UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE GRADO

APLICACIÓN ANDROID PARA LA OBTENCIÓN DE INFORMACIÓN A PARTIR DE FOTOGRAFÍAS

José Manuel Ortiz Cirugeda

Tutor: Luis Fernando Lago Fernández

Julio 2014

RESUMEN

El uso de los dispositivos móviles ha cambiado la forma en que nos comunicamos y conseguimos información. Sin embargo, tener que hacer una búsqueda en Internet de algo con nombre desconocido o extraño, resulta tedioso para el usuario, especialmente si tiene que repetir el proceso varias veces. El problema se puede solucionar incluyendo en los objetos códigos externos que sean posibles de reconocer con la cámara de un *smartphone*, pero esto rompe con estética del objeto o creación y resulta difícil de añadir a elementos ya existentes (como cuadros o libros ya publicados). Poder extraer información a partir de una fotografía es algo que tiene mucho potencial y actualmente ya existen algunas aplicaciones en el mercado que usan estas técnicas de reconocimiento imágenes como *Google Goggles* (1).

En el presente trabajo, se han probado dos técnicas diferentes de reconocimiento de imágenes, usando *OpenCV,* primero en un ordenador y a continuación implementadas en una aplicación para Android. Una de ellas está basada en las bolsas de palabras (BOW) (2). Con ella se extraen características comunes a un conjunto de imágenes que componen una base de datos y se crean palabras visuales con ellas. De esta manera, cada imagen de la base de datos queda clasificada con ciertas palabras de este vocabulario. Extrayendo estas palabras de la imagen que se quiere reconocer, se puede conseguir la más parecida y clasificarla de ese modo.

El otro método implementado se basa en extraer una cadena que sea la que describa la imagen completamente (3). Esta cadena es generada partiendo recursivamente una imagen en dos mitades y comparando el nivel de gris de cada mitad, si la primera mitad es más oscura que la segunda se inserta un 0 en la cadena y un 1 si ocurre lo contrario. La cadena de cada imagen de la base de datos es almacenada para luego ser comparada con la cadena extraída de la imagen que se quiere reconocer. La imagen con la cadena más parecida es la más similar a la imagen de entrada.

Las pruebas desarrolladas muestran que esta segunda estrategia clasifica mejor y más rápido las imágenes de entrada, por lo que es la llevada a la aplicación para dispositivos móviles. En el dispositivo también se lleva a cabo un proceso de homografía (4), que mediante puntos y descriptores ORB (5), puede confirmar la similitud de la fotografía de entrada con la de la base de datos.

Para las pruebas y como aplicación de ejemplo, se ha utilizado una base de datos de cartas del juego *"Magic: The Gathering"* y con otra de portadas de libros, pero el sistema puede funcionar con cualquier otra biblioteca de imágenes bidimensionales.

PALABRAS CLAVE

Reconocimiento de imágenes, OpenCV, Android, puntos característicos, descriptores, bolsa de palabras (BOW)

SUMMARY

The advent of mobile devices has changed the way we communicate and obtain infomation. Nevertheless, finding information about something with an unknown or strange name can result tiresome, especially if the search needs to be repeated several times. This issue can be solved including external codes, recognizable from a smartphone's camera, in the objects to be identified; even though this solution breaks the aesthetics of the creation and is difficult to apply to antique elements (such as ancient paintings or printed books). An alternative solution would be to be able to extract information simply by taking a picture, which has a lot of potential and is starting to be exploited with the creation of some image-recognition apps already present in the market, such as *Google Goggles* (1).

In the present work, two different image-recognition techniques that use *OpenCV* have been tested, firstly in a computer and subsequently implemented in an Android app. One of these is based on the Bag Of Words (BOW) (2) technique. It consists of extracting common features from the pictures in a data base, from which visual keywords are created. Thus, every image in the data base can be classified using some of the keywords in this vocabulary. Extracting the keywords from the image to recognize, it can be classified by searching the most similar one in the data.

The other method implemented in this work is based on extracting a string through which the image is completely described (3). This string is generated

recursively splitting the image in halves and comparing the amount of gray in each half: If the first is darker than the second, a 0 is inserted into the string, while a 1 is inserted otherwise. The string for each of the images in a data base is then stored in order to be compared later with that of a problem picture. The image with the most similar string is chosen as the most similar to the input one. According to the testing carried out in this work, the second strategy classifies input images better and faster than the former method, and therefore it is the latter that has been implemented in the mobile-device app. In the device, a homography process (4) is applied on top of the classifying process, which confirms the similarity between the input and found images using ORB keypoints and descriptors (5).

In order to carry out the testing and for the example app, *"Magic: The Gathering"* Trading Card Game cards and a book cover database have been used, even though the system is designed to work with any set of bidimensional images.

KEYWORDS

Image recognition, OpenCV, Android, keypoints, descriptors, bag of words (BOW)

ÍNDICE

ÍNDICE DE CONTENIDO

1. Introducción

1.1. Objetivos 15

1.2. Estructura del documento 16

2. Estado del arte

2.1. Aplicaciones existentes 18

2.2. Tecnología disponible 21

2.3. OpenCV 28

2.4. Android 29

3. Diseño y desarrollo

3.1. Métodos comunes 35

3.1.1. Corrección de la perspectiva 35

3.1.2. Homografía 38

3.2. Métodos para la clasificación de imágenes 41

3.2.1. Método basado en bolsa de palabras (BOW) 41

3.2.2. Método basado en la comparación de niveles de gris en diferentes secciones de la imagen 44

3.3. Desarrollo de la aplicación móvil 48

4. Pruebas y resultados

4.1. Pruebas de acierto 53

4.1.1. Banco de imágenes de prueba 53

4.1.2. Procedimiento y resultados 54

4.2. Pruebas de eficiencia 63

4.2.1. Banco de imágenes de prueba 63

4.2.2. Procedimiento y resultados 63

4.3. Conclusiones sobre las pruebas 66

4.4. Resultado en dispositivo móvil 68

4.4.1. Implementación en el móvil 68

4.4.2. Pruebas de eficiencia 68

4.4.3. Interfaz y ejemplos 69

5. Conclusiones

5.1. Trabajo desarrollado y futuro 70

5.2. Conclusiones personales 72

6. Referencias

Glosario

Anexos

Anexo A. *MTG Magic Card Recognizer*.....78

Anexo B. Métodos utilizados.....79

ÍNDICE DE TABLAS

TABLA 1. OBJETIVOS DEL PROYECTO DESARROLLADO. 15

TABLA 2. VENTAJAS Y DESVENTAJAS DE EL USO DE JAVA O C/C++ EN UNA APLICACIÓN ANDROID CON *OPENCV*.....31

TABLA 3. CARACTERÍSTICAS DEL ORDENADOR DONDE SE REALIZARON LAS PRUEBAS. 52

TABLA 4. CARACTERÍSTICAS DISPOSITIVO MÓVIL UTILIZADO PARA LAS PRUEBAS. 52

TABLA 5. CARACTERÍSTICAS IMÁGENES DE PRUEBA.....53

TABLA 6. TIEMPOS DE EJECUCIÓN DEL MÉTODO BASADO EN BOW CON 950 PALABRAS. 64

TABLA 7. TIEMPO MEDIO DE EJECUCIÓN DE LA APLICACIÓN EN UN DISPOSITIVO MÓVIL 68

ÍNDICE DE FIGURAS

FIGURA 1. IZQUIERDA: IMAGEN A BUSCAR. DERECHA: IMAGEN DE LA ESCENA. SE PUEDEN OBSERVAR LOS PUNTOS CARACTERÍSTICOS DE CADA IMAGEN (CÍRCULOS DE COLORES) Y COMO SE HAN EMPAREJADO. ADEMÁS SE HA PODIDO HACER LA HOMOGRAFÍA DE LA IMAGEN (RECTÁNGULO VERDE) Y LLEVARLA DE LA FOTO DEL OBJETO A LA FOTO DE LA ESCENA (19). 21

FIGURA 2. RECONOCIMIENTO DE DOS OBJETOS (TREN Y RANA DE JUGUETE) EN UN ESCENARIO CON VARIOS OBJETOS USANDO PUNTOS *SIFT* (20).....22

FIGURA 3. RECONOCIMIENTO DE LOS OBJETOS DE UNA MESA USANDO PUNTOS *ORB* (RUBLEE, 2011) (5) 23

FIGURA 4. ILUSTRACIÓN DE UN ALGORITMO DE CLASIFICACIÓN DE ZAPATOS USANDO BOW. USANDO EL DICCIONARIO CREADO EN (IV) SE EXTRAE UN DESCRIPTOR EN FORMA DE HISTOGRAMA EN ESTE CASO (VIII). (39) 27

FIGURA 6. DIAGRAMA DE LAS FASES PARA EL RECONOCIMIENTO DE IMÁGENES (4). 33

FIGURA 5. DIAGRAMA DE FLUJO PARA LOS PROGRAMAS RECONOCIMIENTO DE IMÁGENES DESARROLLADOS.....33

FIGURA 7. DIAGRAMA DE PASOS DEL FUNCIONAMIENTO DEL ALGORITMO DE CORRECCIÓN DE PERSPECTIVA.....36

FIGURA 8. EJEMPLO DE USO DEL ALGORITMO DE CORRECCIÓN DE PERSPECTIVA. 38

FIGURA 9. DIAGRAMA DE PASOS DEL FUNCIONAMIENTO DEL ALGORITMO DE HOMOGRAFÍA. 39

FIGURA 10. DIAGRAMA DE PASOS DEL FUNCIONAMIENTO DE LA TÉCNICA BASADA EN BOW 41

FIGURA 11. ILUSTRACIÓN DEL ALGORITMO BASADO EN LA COMPARACIÓN DE NIVELES DE GRIS CON UN NIVEL 3 DE RECURSIÓN. EN AZUL, EL ORDEN EN EL QUE EL ALGORITMO ES EJECUTADO. EN ROJO, LA MITAD MÁS OSCURA. EL DESCRIPTOR DE ESTA IMAGEN SERÍA [0, 1, 0, 1, 0, 0, 0]. 45

FIGURA 12. DIAGRAMA DE PASOS DEL FUNCIONAMIENTO DE LA TÉCNICA BASADA EN LA COMPARACIÓN DE NIVELES DE GRIS 46

FIGURA 13. DIAGRAMA DE PASOS DEL FUNCIONAMIENTO DE LA APLICACIÓN.....49

FIGURA 14. ARRIBA: IMÁGENES DE LA BASE DE DATOS. ABAJO: FOTOGRAFÍAS TOMADAS CON EL MÓVIL DE LAS MISMAS CARTAS. 54

FIGURA 15. IZQUIERDA: IMAGEN TOMADA SIN RETOCAR. CENTRO: IMAGEN EN LA QUE SE HA APLICADO EL RECORTE MANUAL. IZQUIERDA: IMAGEN EN LA QUE SE HA APLICADO LA CORRECCIÓN AUTOMÁTICA DE PERSPECTIVA.....55

FIGURA 16. INTERFAZ GRÁFICA DE LA APLICACIÓN. PRIMERA CAPTURA: SELECCIÓN SOBRE LA FUENTE DE LA IMAGEN (CON LA CÁMARA O DESDE LA GALERÍA). SEGUNDA CAPTURA: LA FOTO HA SIDO SELECCIONADA Y SE ESTÁ RECONOCIENDO. TERCERA CAPTURA: LA IMAGEN HA SIDO RECONOCIDA Y SE MUESTRAN SUS DATOS. 69

FIGURA 17. IZQUIERDA: ERROR AL HACER LA FOTOGRAFÍA O AL CORREGIR LA PERSPECTIVA. DERECHA: IMAGEN DE ENTRADA NO RECONOCIDA. 69

1. INTRODUCCIÓN

Durante los últimos años, el desarrollo en el sector móvil y de las telecomunicaciones nos ha permitido acceder a cualquier información disponible en la red a través de nuestros dispositivos móviles con acceso a Internet, ya sea con teléfonos inteligentes (*smartphones*) o con tabletas. Esto ha producido grandes cambios en la forma en la que interactuamos con nuestro entorno, y la forma en la que el entorno se comunica con nosotros.

Hoy en día, es muy normal ver comercios de la calle que nos animan a seguirlos en sus webs o redes sociales, supermercados y vallas publicitarias que nos piden que escaneemos códigos bidimensionales con nuestros *smartphones* para acceder a ofertas, o incluso anuncios de televisión que quieren que "escuchemos" su sintonía con el móvil para acceder a contenidos exclusivos sobre un determinado producto. Estas circunstancias reflejan el hecho de que ya no vale únicamente promocionar un producto o negocio, sino que hay que ofrecer algo más.

Uno de los problemas a los que el cliente o usuario se enfrenta al querer buscar información sobre un producto es el tener que hacer una búsqueda en Internet sobre un objeto cuyo nombre es extraño o incluso desconocido. Esto resulta tedioso para el usuario si tiene que repetir el proceso varias veces en poco tiempo. Es aquí donde entra en acción el reconocimiento de imágenes: ya no hace

falta texto para poder saber datos sobre algo, con una fotografía se puede acceder a ellos.

Estamos familiarizados con aplicaciones en móviles que son capaces de reconocer BIDIs o códigos de barras; o con la famosa aplicación *Google Goggles* (1), que aparte de reconocer códigos binarios, reconoce algunos textos, logotipos de marcas o incluso monumentos y edificios. Con este proyecto se quiere desarrollar una aplicación para móviles con sistema Android para que haciéndole una fotografía a un objeto bidimensional, podamos saber más cosas sobre el mismo o lo que representa. Esto resulta muy interesante para creadores o coleccionistas, que quieren que su pieza no se quede sólo en el plano físico, sino que tenga una apertura al mundo virtual. El hecho de poder reconocer una imagen sin tener que incluir códigos extraños (como BIDIs o códigos de barras), o incluso averiguar información de un cuadro o fotografía antiguos sin tener que añadir ningún elemento moderno (como códigos numéricos), hace de la habilidad de reconocer una imagen algo muy atractivo.

Estas técnicas son muy útiles para, por ejemplo, museos o exposiciones. Un visitante puede tomar una imagen con su dispositivo y conocer más información acerca de la obra: su autor, el año en que se creó, curiosidades sobre esta, etcétera. Otro ejemplo de uso puede ser para coleccionistas o filatélicos, con una fotografía a una pieza pueden saber el valor de un sello o cromo, o saber si este se encuentra en su base de datos o colección. También puede resultar útil para bibliotecas: una sola imagen basta para buscar un libro entre las estanterías.

Para poder llevar a cabo este proceso de reconocimiento, se extrae la parte de interés de la imagen tomada por el usuario y se sacan de ahí las características que la definen. Estas características son comparadas con las imágenes almacenadas en una base de datos y, una vez elegida la más parecida, se muestra información relativa a ella. El diseño implementado permite que toda esta dinámica se haga en el dispositivo, sin tener que acceder a Internet para ello, y devolviendo resultados en un tiempo aceptable para el usuario.

Para las pruebas de este proyecto se ha usado una base de datos del conocido juego de cartas *"Magic: The Gathering"* (edición 2014) (6), además de una colección de portadas de libros que se han recopilado. Es importante tener en cuenta que la técnica que se va a describir es aplicable a muchos tipos de objetos bidimensionales como cromos, cuadros, fotografías o filatelia.

1.1. OBJETIVOS

El principal objetivo del proyecto es crear una aplicación de reconocimiento de imágenes bidimensionales que, dada una fotografía de entrada tomada por el usuario, muestre información sobre esta. Para ello, se utiliza la interfaz para Android de *OpenCV*. En la TABLA 1 se muestran los objetivos concretos del proyecto que se va a desarrollar.

1	La base de datos de imágenes donde se realizarán las búsquedas estará acotada a colecciones de tamaño pequeño o mediano, como por ejemplo: portadas de libros, portadas de discos, cromos o cartas.
2	El proceso de reconocimiento se llevará completamente a cabo en el dispositivo, sin servidores externos implicados en el proceso.
3	El tiempo necesario para reconocer una imagen con el dispositivo móvil debe de estar dentro de unos límites razonables.
4	La aplicación resolverá el problema del reconocimiento y no de la segmentación de la imagen, es decir, la extracción de la región de interés de la fotografía de entrada no debe de ser difícil o costosa, ni la aplicación será capaz de reconocer todos los objetos presentes en una fotografía general.

TABLA 1. OBJETIVOS DEL PROYECTO DESARROLLADO.

1.2. ESTRUCTURA DEL DOCUMENTO

Esta memoria se divide en varios apartados: El primero es una introducción y objetivos donde las características de la aplicación son fijadas y se explica brevemente en qué consiste el reconocimiento de imágenes. En la segunda sección se hace un estudio del estado del arte donde se comenta qué aplicaciones existentes en el mercado utilizan estas técnicas y las definiciones que se utilizarán en el resto del documento, así como cuáles son las diferentes ramas de investigación que se están desarrollando en este campo y un breve repaso a los trabajos de los autores más característicos. En la sección de diseño y desarrollo se detallan los mecanismos utilizados para llevar a cabo el reconocimiento de imágenes, explicando con claridad todos los métodos y algoritmos involucrados en el proceso y cómo han sido utilizados. Al final, se muestran las pruebas realizadas y se analizan sus resultados tanto de eficacia de los algoritmos de reconocimiento de imágenes como de tiempo de ejecución.

2. ESTADO DEL ARTE

En esta sección se hace un recorrido por las aplicaciones ya existentes en el campo del reconocimiento de imágenes. También, se habla de las ramas de investigación más importantes en visión artificial, así como las tecnologías que se utilizan para desarrollar programas que sean capaces de ello.

2.1. APLICACIONES EXISTENTES

Una de las formas más básicas de reconocimiento de imágenes es el reconocimiento de códigos de barras o códigos bidimensionales. Estos códigos se componen por barras o puntos blancos o negros que representan ceros y unos de una cadena. Estos códigos disponen de varios patrones que marcan el inicio o el fin de la cadena o el formato de lo que contiene, por ejemplo, si son caracteres, números o una dirección web.

Fue en los años sesenta cuando se empezaron a utilizar los códigos de barras en Estados Unidos para acelerar el reconocimiento de objetos en almacenes o en industria, pero hasta los años ochenta no tuvo demasiado éxito comercial, cuando se empezó a implantar el sistema en supermercados (7). Ya en los años noventa surgió el código bidimensional (o código *QR*, del inglés *quick response code*), que aunque empezó usándose para medir la velocidad de vehículos, rápidamente se vio que tenía un gran potencial en otros campos, ya que permite

almacenar mucha más información que un código de barras (hasta más de cuatro mil caracteres alfanuméricos en códigos grandes). Además, gracias a su sistema de corrección de errores, se puede restaurar hasta el 30% de dicho código (8).

Con la llegada de los dispositivos inteligentes con cámara, estos códigos empezaron a utilizarse de forma general: ya no sólo en el ámbito de los comercios e industria, sino también en publicidad o para uso personal. Dos de las aplicaciones más populares de reconocimiento de códigos binarios en Android son *QRDroid* de *DroidLa* (9) y el lector QR y de código de barras de *Scanbuy Inc.* (10).

Una de las aplicaciones más populares de reconocimiento de imágenes en dispositivos móviles hoy en día es la aplicación de Google: *Google Goggles* (1). Esta aplicación es capaz de, al hacer una fotografía, reconocer ciertas imágenes que aparecen en ella y hacer una consulta en su motor de búsqueda al respecto. Actualmente, funciona con objetos que son invariables, tales como carátulas de libros y *CDs*, códigos binarios (barras y *QR*), logotipos, textos, obras de arte, monumentos y lugares importantes. No funciona tan bien con cosas que por su naturaleza son cambiantes (como animales o plantas) o muy parecidos entre sí (muebles o coches) (11).

Para el reconocimiento de imágenes con *Google Goggles*, la foto tomada por el usuario se sube a la nube de Google y es en los servidores de esta empresa donde se extrae la información de ella y se pasa por su sistema entrenado para el reconocimiento de imágenes (12). La principal pega de este mecanismo reside en el gran gasto de datos móviles que utiliza la aplicación para reconocer una imagen,

ya que, aunque se reduce el tamaño de la misma antes, hay que subirla a través de Internet a sus servidores, además, si la aplicación no tiene acceso a la red, no funciona. Las ventajas que posee Google son la gran cantidad de ordenadores dedicados a este proceso que tiene y el gran banco de imágenes del que dispone. Gracias a esto, les es posible devolver una respuesta acorde con la imagen en pocos segundos.

Actualmente, existen multitud de aplicaciones para móviles que son capaces de reconocer códigos de barras y códigos bidimensionales, aunque no tantas de reconocimiento de imágenes no binarias. Aparte de las *Goggles* de *Google*, existen otras aplicaciones parecidas: como *CamFind* de *Image Searcher Inc*. (13). Además de otras más especificas como *NARUTO CARD SCANNER* (14) o *POWER RANGERS SCANNER* (15) de *Bandai America*, que reconocen cartas de las colecciones de dichas series. Todas estas aplicaciones necesitan acceso a la red para poder funcionar.

Comercialmente existen varias empresas, como *Moodstocks.com* (16) o *TinEye.com* (17) que ofrecen *APIs* (*Application Programming Interface*) que dada una imagen, son capaces de cotejarla en sus servidores y bases de datos y devolver un resultado al usuario. Sin embargo, estas empresas no detallan el mecanismo que usan para dicho proceso.

2.2. TECNOLOGÍA DISPONIBLE

En el ámbito académico, una de las técnicas más usadas para el reconocimiento de imágenes está basada en el uso de puntos característicos o *keypoints* (4) (18). El objetivo es reconocer puntos y características invariables en la imagen. En particular, cuando se trata de objetos planos, el planteamiento está basado en encontrar una proyección (homografía) que lleve los puntos de una imagen a los de otra (4).

Un ejemplo de esto se muestra en la FIGURA 1. El objetivo es buscar la imagen de la izquierda (objeto) en la imagen de la derecha (escena).

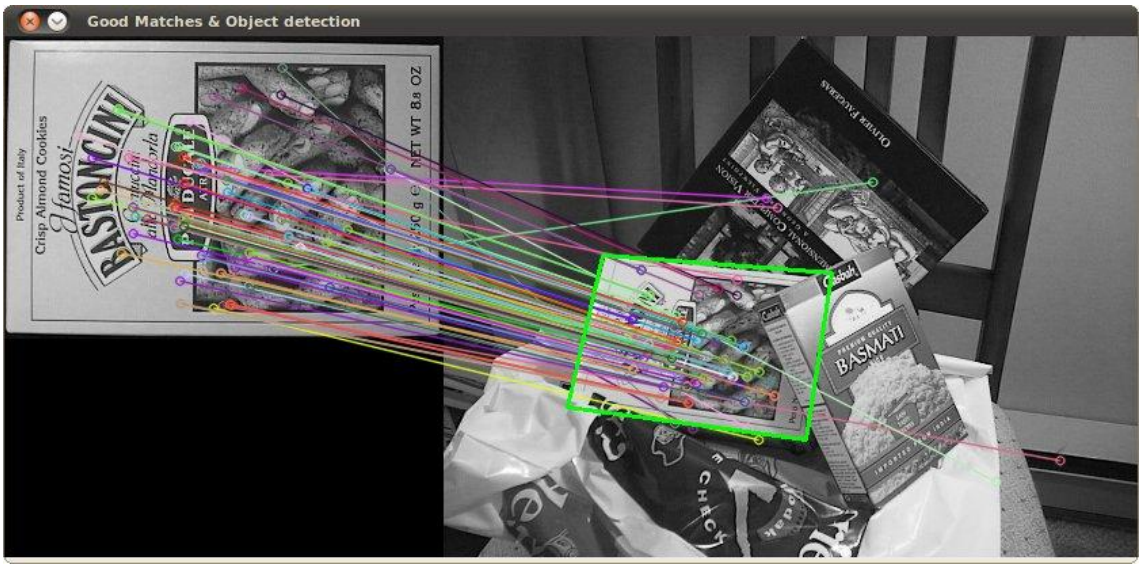


FIGURA 1. IZQUIERDA: IMAGEN A BUSCAR. DERECHA: IMAGEN DE LA ESCENA. SE PUEDEN OBSERVAR LOS PUNTOS CARACTERÍSTICOS DE CADA IMAGEN (CÍRCULOS DE COLORES) Y COMO SE HAN EMPAREJADO. ADEMÁS SE HA PODIDO HACER LA HOMOGRAFÍA DE LA IMAGEN (RECTÁNGULO VERDE) Y LLEVARLA DE LA FOTO DEL OBJETO A LA FOTO DE LA ESCENA (19).

Seguidamente se describen los elementos necesarios para este enfoque.

1. Puntos característicos y descriptores

Un punto característico es un lugar concreto de una imagen que contiene una propiedad que la define. Esto puede ser, por ejemplo, esquinas, bordes, formas u otros elementos que representen una particularidad de la imagen. Existen multitud de tipos de puntos característicos, entre ellos *SIFT* (20) (ver FIGURA 2) y *SURF* (21), que son los más extendidos, y otros como los *BRIEF* (22) o *KAZE* (23) que por sus características, son menos eficientes en su funcionamiento, aunque usados en ocasiones concretas.

Una vez extraídos los puntos característicos, se genera un descriptor para cada uno de ellos. Los descriptores son estructuras que contienen varios datos sobre el punto y su entorno, para poder definir un contexto para el mismo. Idealmente, estos descriptores deben ser invariantes a la escala, traslación o rotación de la imagen, así como cambios de iluminación o ruido (4).

FIGURA 2. RECONOCIMIENTO DE DOS OBJETOS (TREN Y RANA DE JUGUETE) EN UN ESCENARIO CON VARIOS OBJETOS USANDO PUNTOS *SIFT* (20)

Otro de los tipos de punto y descriptor más conocidos son los tipo *ORB* (5). Rublee explica como la combinación de dos tipos de puntos diferentes (*FAST* y *BRIEF*) en un tipo diferente llamado *ORB* (*Oriented FAST and Rotated BRIEF*) podía plantarle cara a los puntos *SIFT* y *SURF* en cuestión de eficiencia y precisión (ver FIGURA 3).

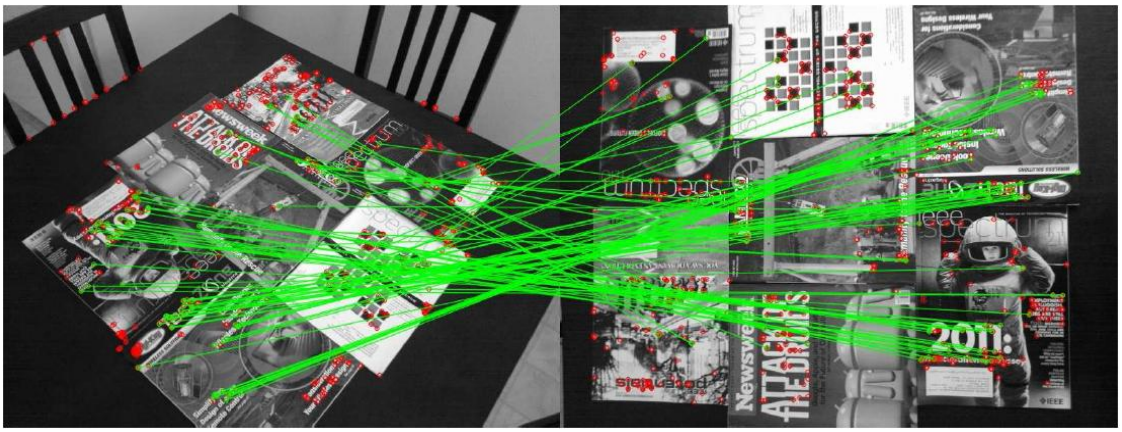


FIGURA 3. RECONOCIMIENTO DE LOS OBJETOS DE UNA MESA USANDO PUNTOS *ORB* (RUBLEE, 2011) (5)

En este proyecto se han utilizado los puntos y descriptores ORB, ya que ofrecen ciertas ventajas:

1. Son más rápidos en su procesamiento, ya que sus descriptores son binarios.
2. En cuestión de precisión, son equivalentes a SIFT y SURF, que son los más extendidos (5).
3. ORB es de uso libre, mientras que SIFT y SURF operan bajo derechos de autor.

2. Emparejamiento de puntos

El emparejamiento de puntos, es el proceso mediante el cual los descriptores de dos imágenes son comparados y unidos en parejas. Existen varios métodos para este proceso, el más sencillo es el del vecino más próximo, dónde se calcula la distancia entre los descriptores y se emparejan con el más cercano. El problema de este método es que siempre se emparejan los puntos, independientemente de si pertenecen o no a la imagen. Para solucionar esto, Lowe aplica un límite de lejanía, esto es, si dos puntos emparejados están lo suficientemente lejos el uno de otro, son eliminados del recuento de parejas (20).

Esta técnica descrita es la más básica a la hora de emparejar y puede resultar muy lenta, ya que se comparan todos los puntos de la imagen de entrada con todos los puntos de las imágenes de la base de datos. Este proceso se puede acelerar usando técnicas usadas en inteligencia artificial y minería de datos como los *K-means* (ver Anexo B. Métodos utilizados), árboles de decisión y bolsas de palabras (2).

3. Homografía

Una vez se tienen los puntos emparejados, se busca una homografía que transforme los puntos de la primera imagen en la

segunda. Una homografía es una transformación proyectiva entre dos figuras geométricas planas. En general se consideran que dos imágenes son equivalentes si existe dicha transformación. Para determinar los parámetros de la transformación, normalmente se utilizan sólo unas pocas parejas de puntos. La transformación resultante se aplica al resto de puntos, comparando el punto transformado con el punto emparejado.

La homografía debe ser independiente del tamaño, localización y rotación del objeto de la escena. Este procedimiento es usado para comprobar qué cantidad de puntos de la imagen de entrada pertenecen a la imagen de la base de datos y si estos coinciden. Si se dispone de las suficientes parejas buenas, la imagen ha sido reconocida

Sam S. Tsai en 2010 (24) ya realiza un estudio sobre la eficiencia del reconocimiento de imágenes en dispositivos móviles. En su artículo propone la extracción de los puntos característicos y sus descriptores en el propio dispositivo, para luego mandarlos a un servidor en Internet que sea el que se encargue de realizar la búsqueda en la base de datos y ofrecer de vuelta una respuesta al usuario.

Antonio Torralba (3) habla de cómo unos cuantos bits de una imagen pueden bastar para definirla, sin necesidad de tener que reconocer la imagen

entera. Para ello utiliza descriptores más básicos que son más rápidos o mejores para dispositivos con poco almacenamiento.

La técnica de bolsa de palabras (BOW, del inglés *Bag Of Words*), es de las más extendidas en el ámbito de la categorización de imágenes usando puntos característicos (18), pues es un procedimiento muy potente a la hora de clasificar nuevos elementos según sus atributos. Este método proviene del área de clasificación de textos: se hace un estudio de cuantas veces se repite cada palabra en muchos textos de diferentes clases, con ello se saca una estadística que permite clasificar un texto nuevo en alguna categoría. Por ejemplo, si se repiten mucho las palabras "euro", "cuenta bancaria" o "inflación", el texto de entrada será clasificado en el ámbito económico y no en el de recetas de cocina. Esta táctica también se puede aplicar a imágenes, pero en vez de hacer un estudio acerca de qué palabras se repiten más, se tiene en cuenta qué puntos característicos son más frecuentes (25).

Para que esto sea posible, primero hay que construir un vocabulario con unas palabras o características comunes que nos permitan definir los objetos y diferenciarlos entre ellos (ver FIGURA 4). Una vez que disponemos de este vocabulario, se crea un diccionario donde asignamos cada objeto a una definición o conjunto de características. Cuando el diccionario ha sido creado, para poder clasificar una imagen hay que caracterizarla usando dicho diccionario, es decir, crear un histograma con las apariciones de cada "palabra". Calculando las distancias entre histogramas, conseguimos encontrar las imágenes más parecidas.

Esta búsqueda puede ser muy costosa en tiempo y recursos, por ello, algunos autores han propuesto el uso de un árbol de decisión para acelerar las búsquedas (2). Esto sirve para poder ir eliminando posibles resultados en base a características que diferencian mucho los objetos. Por ejemplo, si un objeto tiene una característica "ruedas", podemos estar seguros de que no va a ser clasificado en la categoría "frutas" o "edificios", ahorrándose así el tener que buscar el objeto en estas categorías. Al final de este proceso, se encuentra una lista mucho más corta con las clases que cumplen varias características analizadas para poder devolver unos resultados más concretos.

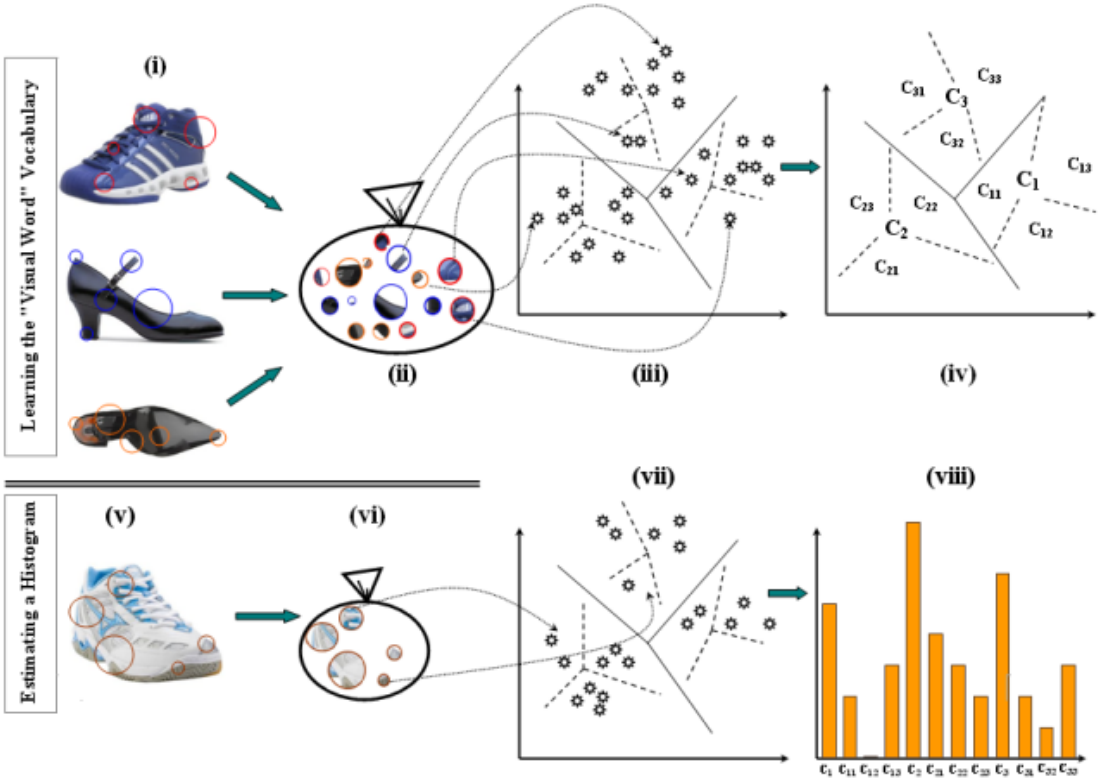


FIGURA 4. EJEMPLO DE UN ALGORITMO DE CLASIFICACIÓN DE ZAPATOS USANDO BOW. USANDO EL DICCIONARIO CREADO EN (IV) SE EXTRAE UN DESCRIPTOR EN FORMA DE HISTOGRAMA EN ESTE CASO (VIII). (39)

2.3. OPENCV

Para poder hacer programas utilizando los procedimientos descritos en este apartado, se utiliza una librería de código abierto llamada *OpenCV* (*Open Computer Vision*) (26). Ésta es una librería escrita en los lenguajes de programación *C* y *C++* que actualmente está disponible tanto para *Linux*, como *Windows* y *Mac*, además de ofrecer varias interfaces de desarrollo para lenguajes como *Java*, *Python*, *Ruby* o *Matlab*. En los últimos años, también han lanzado su compatibilidad con dispositivos *Android* (2010) e *iOS* (2012) (27).

Esta librería nació en los laboratorios Intel a finales de los años noventa y desde entonces se encuentra en multitud de aplicaciones, como por ejemplo sistemas de seguridad y control (26). Esta gran popularidad se debe a su licencia de carácter libre, que permite a desarrolladores utilizarla y modificarla como deseen, tanto para aplicaciones académicas como comerciales, sin tener la obligación de liberar el código de estos nuevos programas.

OpenCV contiene más de quinientas funciones para el ámbito de la visión artificial, incluyendo reconocimiento y tratamiento de imágenes o reconocimiento facial. *OpenCV* pretende ser sencillo de utilizar y de aprender, además de muy eficiente, ya que las librerías escritas en *C* y *C++* se han optimizado para ser capaces de funcionar en aplicaciones de tiempo real.

Concretamente, para este proyecto se utilizan sobre todo las funciones relativas al tratamiento de imágenes (como pasar de una foto en color a otra en

escala de grises) y funciones para la extracción de puntos característicos, así como sus descriptores, emparejamientos y homografía. También, y como se detallará más adelante, se utilizan funciones de aprendizaje automático incluidas por defecto en esta librería, como la bolsa de palabras y los vecinos más próximos.

2.4. ANDROID

Android, es un sistema operativo basado en Linux diseñado especialmente para dispositivos con pantallas táctiles como *smartphones* y *tablets*. En 2005, Google se hizo cargo de este sistema y quiso potenciar el desarrollo de *hardware* y *software* en el campo de las telecomunicaciones (28). A fecha de hoy, este sistema operativo se encuentra en la versión 4.4. (*KitKat*), y se espera que en los próximos meses lleguen nuevas actualizaciones (29).

Uno de los objetivos de *Google*, como de las otras empresas que componen la *Open Handset Alliance* con *Android*, era abrir el sistema a los desarrolladores para que se pudiesen desarrollar aplicaciones y nuevas actualizaciones de la manera más sencilla posible. Actualmente, las aplicaciones *Android* están escritas en *Java* usando la *API* que proporciona *Google* a los desarrolladores (30). Esta *API* no sólo incluye la librería estándar de *Java*, si no varias librerías de terceros que resultan muy útiles para el desarrollo de aplicaciones: como para crear bases de datos en *SQLite* o comunicación por red. También es posible añadir librerías externas a las aplicaciones de modo que sea posible implementar funcionalidades extra a las aplicaciones (31).

Aunque el desarrollo de la aplicación central se hace en *Java*, es posible añadirle a la aplicación código en *C* o *C++* utilizando el *NDK* (*Native Development Kit*) que proporciona *Android*. Esto resulta muy interesante para el desarrollo de juegos o aplicaciones de tiempo real, ya que el código compilado de *C* o *C++* se ejecuta más eficientemente que el *bytecode* de Java (ver TABLA 2).

Como se ha mencionado anteriormente, *OpenCV* es multiplataforma y nos permite desarrollar programas en múltiples lenguajes. Para el desarrollo de aplicaciones *Android*, *OpenCV* da la oportunidad de hacerlo de dos maneras distintas: Usando la interfaz de *OpenCV* para *Java-Android* o desarrollando el código en *C/C++* y después enlazándolo con el resto del código del proyecto de la aplicación. Para poder disfrutar de las funciones de *OpenCV* en el dispositivo Android, es necesario tener instalada por separado la aplicación propia de *OpenCV*, que es la que contiene las funciones que se van a usar. Además, hay que hacer una comprobación de si está instalada la aplicación en el móvil y abrirla internamente desde nuestra aplicación antes de usar cualquiera de las funciones de dicha plataforma.

	Interfaz *Java - Android*	***C/C++***
Ventajas	- Configuración simple, sólo hay que referenciar la librería. - Sintaxis sencilla, como *Java.*	- Muy eficiente para el desarrollo de grandes aplicaciones o aplicaciones en tiempo real.
Desventajas	- No tan eficiente como con *C* o *C++*, aunque la interfaz Java sólo es una llamada a la función nativa.	- Configuración del *IDE* y de la aplicación más complicada. - Depuración más complicada, pues no se puede pasar de depurar *Java* a *C/C++*.

TABLA 2. VENTAJAS Y DESVENTAJAS DE EL USO DE JAVA O C/C++ EN UNA APLICACIÓN ANDROID CON *OPENCV*.

3. DISEÑO Y DESARROLLO

En esta sección se va a comentar en profundidad cómo se han desarrollado dos alternativas diferentes para el reconocimiento de imágenes: usando *BOW* y usando otro método basado en la comparación de niveles de gris en diferentes secciones de la imagen. *BOW* se basa en la extracción de características comunes de las imágenes creando un diccionario, donde después se hace una búsqueda con las palabras detectadas en la imagen que se quiere reconocer o clasificar. La técnica basada en la comparación de niveles de gris es un método que hemos desarrollado en el que la imagen es transformada a escala de grises, se parte por la mitad y se calcula qué mitad es más oscura. Dependiendo de qué mitad sea la más oscura, se añade un 0 ó un 1 a una cadena. Las mitades se vuelven a partir de nuevo recursivamente y se repite el proceso varias veces, consiguiendo una cadena binaria como descriptor global de la imagen. Finalmente, se calculan las distancias de Hamming entre descriptores para conseguir la imagen más cercana a la de la consulta.

Ambos métodos comparten dos partes comunes que son utilizadas para mejorar los resultados del reconocimiento: la corrección de perspectiva y la homografía (ver FIGURA 5). Primero, el usuario toma una fotografía del objeto que quiere reconocer. De esa fotografía se extrae la región de interés y es la que se pasa a alguno de los algoritmos mencionados. El algoritmo devuelve una lista de

posibles imágenes cercanas, con las que se hace una homografía que concrete el resultado.

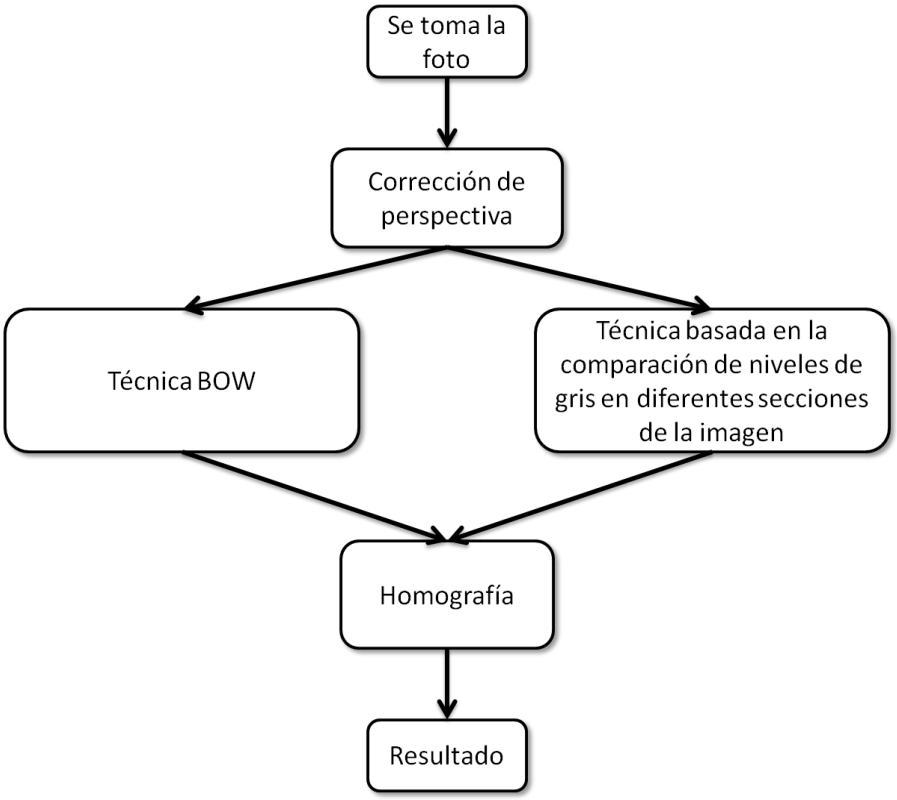


FIGURA 5. DIAGRAMA DE FLUJO PARA LOS PROGRAMAS RECONOCIMIENTO DE IMÁGENES DESARROLLADOS.

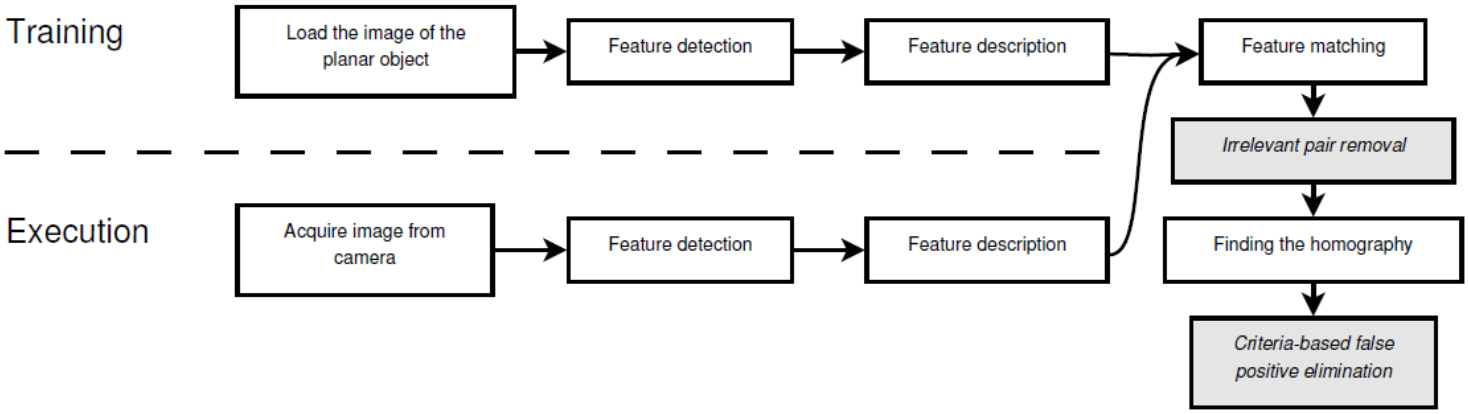


FIGURA 6. DIAGRAMA DE LAS FASES PARA EL RECONOCIMIENTO DE IMÁGENES (4).

Como se puede ver en la FIGURA 6, generalmente existen tres fases en el reconocimiento de imágenes (4): La fase de entrenamiento, en la que el sistema se encarga de extraer los puntos característicos y descriptores de los elementos que componen la base de datos; fase de ejecución, que es en la que se obtienen los puntos característicos y descriptores de la imagen de entrada; y por último, la fase de reconocimiento, que es la encargada de efectuar el emparejamiento de los descriptores de la imagen de entrada y hallar una homografía válida.

3.1. MÉTODOS COMUNES

3.1.1. CORRECCIÓN DE LA PERSPECTIVA

Con el objetivo de simplificar el reconocimiento y clasificación de las imágenes, se implementa un algoritmo de corrección de perspectiva, esto es, detectar el objeto, en este caso bidimensional y rectangular, en la imagen de entrada y transformarlo a otra donde sus esquinas estén en las mismas esquinas de la imagen.

Para simplificar el algoritmo, suponemos que el objeto de interés es fácil de segmentar en la imagen de entrada. Para ello, exigiremos que la imagen tomada por el usuario esté hecha sobre fondo claro y sin elementos extraños alrededor. La segmentación de imágenes no forma parte de este proyecto.

Este algoritmo ayuda a mejorar la precisión del reconocimiento, sobre todo para el método basado la comparación de niveles de gris. Teóricamente, para el método basado en BOW, esto no sería necesario, pero como se ve en las pruebas realizadas (Sección 4), la clasificación mejora también. Esto se debe a que parte de las características o puntos encontrados en la fotografía de entrada se encuentran en el límite del objeto o en el fondo, y en la base de datos esto no ocurre, pues la información de la imagen llega hasta los bordes.

3.1.1.1. Funcionamiento

En la FIGURA 7 se pueden ver los pasos tomados por el algoritmo de corrección de perspectiva para extraer la región de interés de la imagen de entrada. En la FIGURA 8 se puede ver un ejemplo de cómo funciona este algoritmo.

FIGURA 7. DIAGRAMA DE PASOS DEL FUNCIONAMIENTO DEL ALGORITMO DE CORRECCIÓN DE PERSPECTIVA.

1. Simplificación de la imagen: La imagen de entrada es reducida de tamaño y pasada a escala de grises para un mejor tratamiento de la misma.
2. Detección de bordes y contornos: La imagen en escala de grises es difuminada y pasada al algoritmo de Canny (32). Este algoritmo detecta bordes en la imagen y devuelve una matriz binaria con éstos. Sobre esta matriz, se pueden usar funciones propias de *OpenCV* que son capaces de detectar y dibujar los contornos encontrados en la imagen. De nuevo, se repite el proceso difuminando, aplicando Canny y las funciones de detección y dibujo de contornos, pero esta vez buscando sólo el borde más externo. Repetir este proceso sobre la imagen ayuda a eliminar ruidos en los bordes y conseguir un mejor contorno.
3. Detección de esquinas: Se usa la función de Harris (33) para la detección de esquinas sobre la matriz que se devuelve al detectar los contornos. Esta función rellena aquellos píxeles que más probabilidades tienen de

ser una esquina con un valor más alto que los que no. Estos valores son normalizados en escala de grises, para asignarles un valor entre 0 (negro) y 255 (blanco).

4. Selección y ordenación de esquinas: De la matriz con los valores normalizados, se seleccionan sólo aquellas que tengan un valor superior a un límite definido, para eliminar posible ruido y bordes mal definidos. De estos puntos seleccionados, se eligen aquellos que no tengan esquinas alrededor, para quedarnos sólo con las cuatro esquinas que tiene la imagen a reconocer. Con estos cuatro puntos seleccionados se calcula el centro y se ordenan según las esquinas sean superiores o inferiores a ese centro, y estén a la izquierda o derecha de él.
5. Transformación de perspectiva: Con los puntos colocados con cada una de las cuatro esquinas, se crea una matriz de transformación. Finalmente, se convierte la imagen de entrada a una nueva imagen usando la matriz de transformación que proyecta las esquinas localizadas en la imagen de salida.

Si los bordes no han sido bien detectados, se han encontrado más o menos de cuatro esquinas en la imagen, o las cuatro esquinas encontradas no son lo suficientemente buenas, la función devuelve un error.

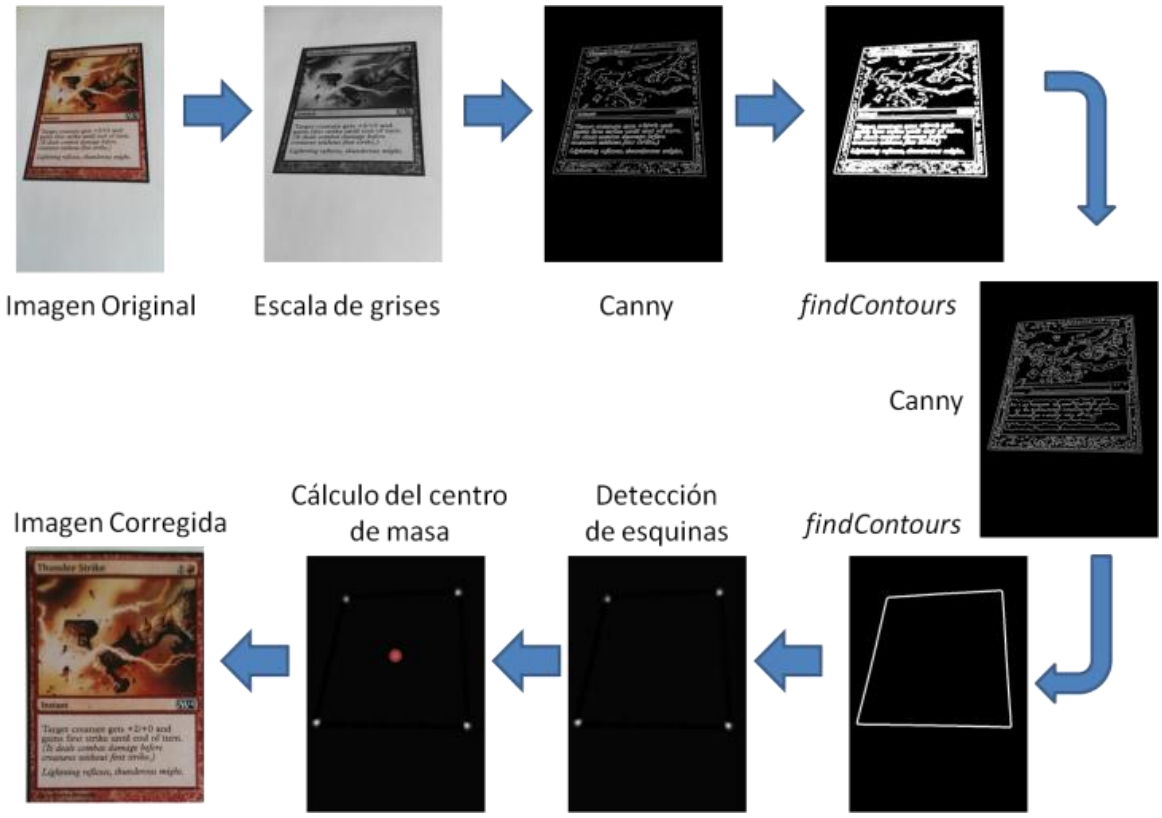


FIGURA 8. EJEMPLO DE USO DEL ALGORITMO DE CORRECCIÓN DE PERSPECTIVA.

3.1.2. HOMOGRAFÍA

Este paso, también común a las dos aproximaciones para la clasificación de imágenes, sirve para confirmar que entre las imágenes más parecidas encontradas por estos algoritmos se encuentra la imagen buscada. Esta función busca la homografía de una imagen sobre otra y calcula el número de emparejamientos de puntos que son válidos en el reconocimiento, llamados *inliers*, así como el número de falsos positivos, llamados *outliers*.

Esta función es llamada desde el programa principal, que mediante un bucle, va comparando la imagen de entrada con la lista de las más parecidas según la alternativa utilizada. El programa, se queda con la imagen que mejor porcentaje de *inliers* presenta, para así confirmar el resultado del método aplicado.

3.1.2.1. Funcionamiento

En la FIGURA 9, se pueden observar todos los pasos tomados por el algoritmo para hallar la homografía.

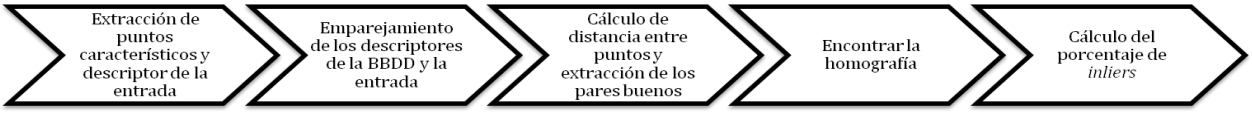


FIGURA 9. DIAGRAMA DE PASOS DEL FUNCIONAMIENTO DEL ALGORITMO DE HOMOGRAFÍA.

1. Extracción de puntos característicos y descriptores de la entrada: Para detectar los puntos característicos de la entrada, se crea un detector de puntos tipo ORB. Este detector recibe la imagen y devuelve una lista con los puntos detectados. De la misma manera, se crea un extractor de descriptores tipo ORB, que junto con la imagen y los puntos característicos, devuelve una lista de descriptores.
2. Emparejamiento de los descriptores de la base de datos y la entrada: Los puntos característicos y los descriptores de las imágenes de la base de datos no es necesario calcularlos cada vez, ya que han sido almacenados previamente en dicha base de datos. Es el programa principal el que se encarga de comparar las imágenes más parecidas según el algoritmo utilizado con la imagen de entrada. Por cada imagen que se le pasa a esta función, se emparejan los descriptores usando un algoritmo de fuerza bruta y la distancia de Hamming. El algoritmo empareja cada punto con el que menor distancia de Hamming presente, es decir, con el que menos distancia de palabra tenga. Esto, que a priori puede parecer ineficiente, es más eficaz

para emparejar descriptores de naturaleza binaria como los ORB, ya que evita cálculos innecesarios (5).

3. Cálculo de la distancia entre puntos y extracción de los pares buenos: Ahora se recorre toda la lista de emparejamientos y se encuentra la distancia mínima entre los descriptores que la componen. Sólo aquellas parejas que tengan una distancia menor que el triple de la distancia mínima son considerados como pares válidos.
4. Encontrar la homografía: De cada par válido se extraen los puntos característicos tanto de la consulta como de la base de datos. Estos puntos son los utilizados por la función para encontrar homografías de *OpenCV* junto con el algoritmo de *RANSAC* (34). Con este algoritmo se seleccionan varias parejas de puntos; se calcula la transformación entre esos puntos y se comprueba que la distancia entre ellos no sea mayor que una tolerancia definida. Si un punto es considerado válido, se llama *inlier*. Si el número de *inliers* encontrados es mayor que un límite determinado, se estima el modelo para todos los puntos, si no, se repite el proceso un máximo definido de veces.
5. Cálculo del procentaje de *inliers*: Con la cantidad de *inliers* encontrados y dividiéndolo entre el número de parejas válidas encontradas, podemos conseguir un número entre 0 y 1 que representa la fiabilidad de esta homografía.

3.2. MÉTODOS PARA LA CLASIFICACIÓN DE IMÁGENES

En este apartado se van a describir los dos módulos desarrollados para el reconocimiento de imágenes: uno basado en bolsa de palabras y otro basado en la comparación de niveles de gris.

3.2.1. MÉTODO BASADO EN BOLSA DE PALABRAS (BOW)

3.2.1.1. Funcionamiento

En la FIGURA 10, se muestran los pasos que toma este algoritmo. Estos pasos son desarrollados a continuación.

FIGURA 10. DIAGRAMA DE PASOS DEL FUNCIONAMIENTO DE LA TÉCNICA BASADA EN BOW

1. Extracción de las palabras para crear un diccionario: Se recorre toda la base de datos de imágenes extrayendo los descriptores de cada una. Estos descriptores son añadidos a una clase propia de *OpenCV* llamada *BOWKMeansTrainer*. Una vez que todos los descriptores son añadidos a esta clase, se utiliza el método *cluster* de esta misma, que se encarga de aplicar el algoritmo *K-means* a los descriptores de las imágenes, haciendo de cada *cluster* una palabra. Estas palabras son devueltas en forma de matriz para utilizarlas en el siguiente paso.
2. Entrenar datos para asociar las palabras a etiquetas: En este paso se utiliza otra clase de *OpenCV* llamada *BOWDescriptorExtractor*, que se

encarga de caracterizar las imágenes de la base de datos con las palabras del diccionario creado. Estos descriptores tipo BOW se van añadiendo a una matriz y, paralelamente, creando otra matriz con la etiqueta que se corresponde con la imagen en la misma posición.

3. Obtener imagen de entrada con la perspectiva correcta y extracción de sus puntos característicos: Ahora se toma la imagen de entrada y se corrige su perspectiva como se explicó en la sección anterior. Además, se extraen los puntos característicos, que serán utilizados en el siguiente paso.
4. Descriptor de la imagen de entrada según el diccionario de BOW: Usando la clase *BOWDescriptorExtractor* del paso 2, que esta entrenada con los datos del diccionario creado en el paso 1, se consigue el descriptor tipo BOW de la imagen de entrada. Sólo es necesario pasar la imagen de entrada y los puntos característicos del paso 3 a esta clase para tener el descriptor deseado.
5. Uso de KNN para conseguir las imágenes más cercanas: Para conseguir una lista con las imágenes más parecidas a la de entrada, se utiliza un algoritmo para calcular los vecinos más próximos, que es inicializado con los datos entrenados y las etiquetas del paso 2. Usándolo podemos averiguar las *K* imágenes más cercanas, ya que devuelve ordenadas de más cercana (o parecida) a más lejana (o diferente) las etiquetas de las imágenes de la base de datos así como la distancia calculada. Usando

esta lista de imágenes, podemos aplicar la homografía a cada una de ellas para poder averiguar el resultado final.

3.2.2. MÉTODO BASADO EN LA COMPARACIÓN DE NIVELES DE GRIS EN DIFERENTES SECCIONES DE LA IMAGEN

A lo largo de este documento se ha hablado sobre los descriptores basados en puntos característicos; a estos descriptores se les llama locales, pues describen una zona limitada de una imagen. También existen otros descriptores llamados globales, que lo que describen es la imagen entera (4). La alternativa que se va a desarrollar a continuación utiliza este tipo de descriptores para definir una imagen.

Esta táctica consiste en pasar la imagen a escala de grises y recursivamente, ir partiendo por la mitad la imagen (ver FIGURA 11). Por cada mitad generada, se calcula la media de las intensidades de gris de los píxeles de esa mitad. Según qué mitad sea más oscura, se añade un 0 ó un 1 a una cadena. Recursivamente, se vuelve a partir cada mitad en dos mitades más y a añadir un 0 ó un 1 a la cadena. Este proceso se repite un número determinado de recursiones para poder generar un descriptor global.

La ventaja principal de este método es la sencillez del mismo para conseguir un descriptor, así como el ahorro de espacio de almacenamiento que esto supone, ya que con una cadena binaria de unos pocos bits, es posible definir una imagen entera. Antonio Torralba habla en su artículo de 2006 (3) de que unos pocos bits son necesarios para poder ser capaces de clasificar una imagen.

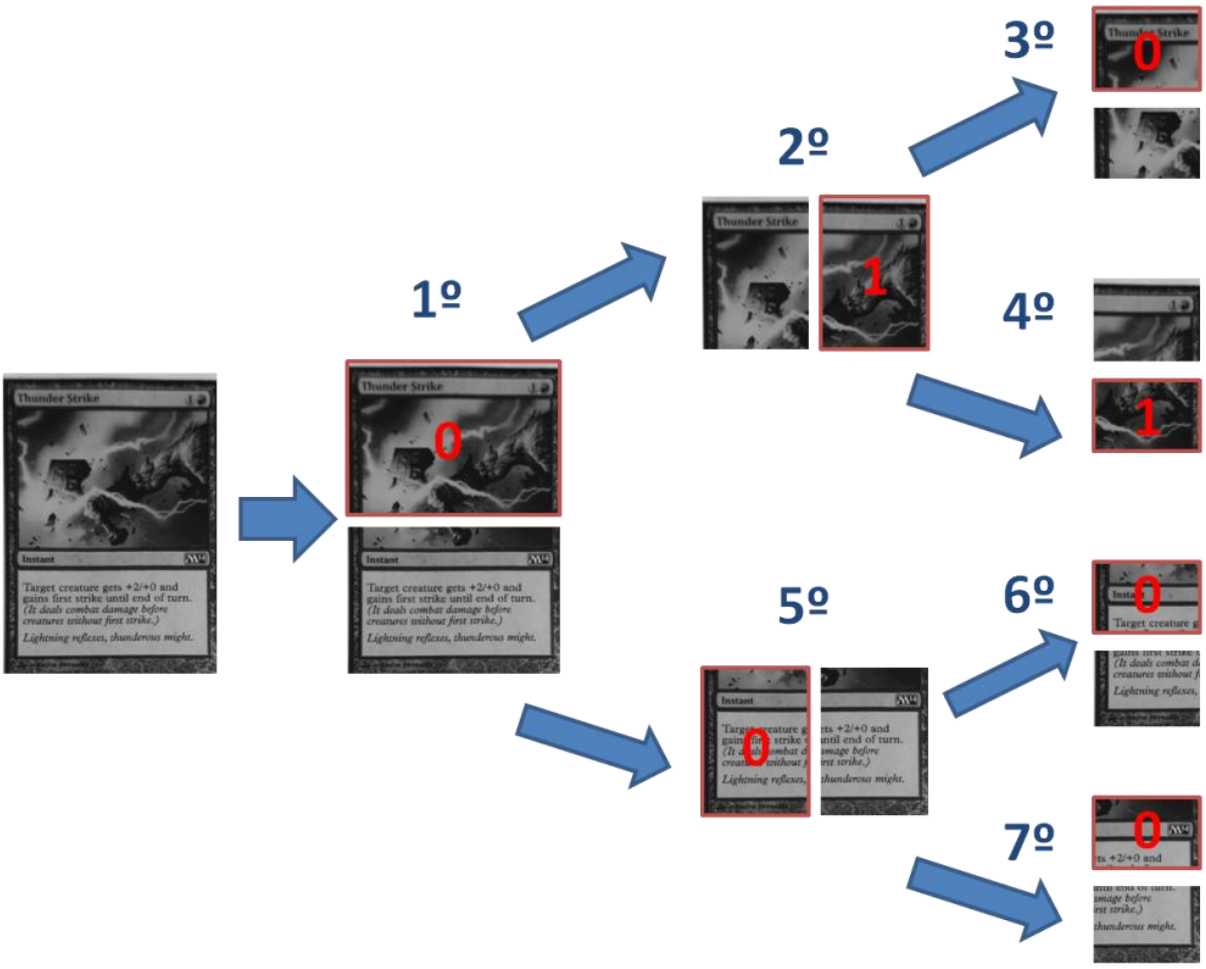


FIGURA 11. EJEMPLO DEL ALGORITMO BASADO EN LA COMPARACIÓN DE NIVELES DE GRIS CON UN NIVEL 3 DE RECURSIÓN. EN AZUL, EL ORDEN EN EL QUE EL ALGORITMO ES EJECUTADO. EN ROJO, LA MITAD MÁS OSCURA. EL DESCRIPTOR DE ESTA IMAGEN SERÍA [0, 1, 0, 1, 0, 0, 0].

3.2.2.1. Funcionamiento

En la FIGURA 12 se muestran los pasos tomados para ejecutar este algoritmo. Seguido de la explicación detallada de estos.

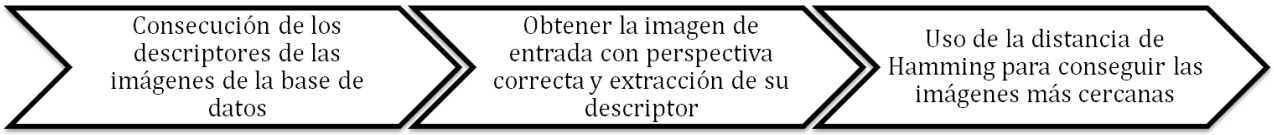


FIGURA 12. DIAGRAMA DE PASOS DEL FUNCIONAMIENTO DE LA TÉCNICA BASADA EN LA COMPARACIÓN DE NIVELES DE GRIS

1. Consecución de los descriptores de las imágenes de la base de datos: Se recorre toda la base de datos de imágenes calculando el descriptor de cada una. Para ello, se pasa la imagen en escala de grises a la función *getImageDescriptor* que se ha desarrollado. Esta función recibe la matriz con la imagen en escala de grises y un entero que es la profundidad de la recursión que se va a utilizar. Esta función llama a la función recursiva *getImageDescriptorRec*, que tiene como parámetros la imagen, y dos enteros, que es el nivel de recursión que se tiene que hacer y el nivel al que se tiene que llegar. La imagen es partida por la mitad con la función desarrollada *cropImage*, que usa la clase *Rect* de *OpenCV* para conseguir una zona determinada de una imagen que, en este caso, es la mitad partiendo por el lado más largo de la imagen. Las dos mitades son almacenadas en una lista que es devuelta a la función anterior. Se pasa cada mitad de la imagen a la función que calcula la media aritmética de *OpenCV*, que devuelve la media de intensidad de gris de cada pixel. Si la primera mitad es más oscura, se añade un 0 a la

cadena de salida, si no, se añade un 1. Se aumenta en una unidad el nivel actual de recursión y se llama de nuevo a la misma con las dos mitades. Cuando se alcanza el nivel requerido de recursión, la función devuelve una lista con los ceros y unos que componen el descriptor. Cada descriptor de cada imagen es almacenado en una lista junto con el identificador de ésta.

2. Obtener la imagen de entrada con perspectiva correcta y extracción de su descriptor: Usando las funciones explicadas anteriormente, se corrige la perspectiva de la imagen de entrada y se calcula su descriptor como se indica en el primer paso.
3. Uso de la distancia de Hamming para conseguir las imágenes más cercanas: El descriptor de la imagen de entrada es comparado con cada descriptor de la base de datos y se almacena el resultado del cálculo de la distancia de Hamming en una lista. Cuando ya están todos calculados, se ordena esta lista y se seleccionan las imágenes más cercanas por este método.

3.3. DESARROLLO DE LA APLICACIÓN MÓVIL

Para decidir qué método será el implementado para dispositivos Android, se realizarán varias pruebas de acierto y eficiencia en un ordenador de sobremesa. Estas pruebas están detalladas en la sección 4 de este documento.

La aplicación móvil que se va a desarrollar contará con los mismos mecanismos descritos en apartados anteriores: la corrección de perspectiva, el mecanismo utilizado para la clasificación y la homografía para comprobar este resultado.

El desarrollo de una aplicación Android implica cambios en la forma en la que se utilizan los programas desarrollados, ya que va a hacer falta un mecanismo para captar imágenes con la propia cámara del dispositivo o seleccionarlas desde la galería de fotos del mismo. También, hay que desarrollar una interfaz gráfica que permita utilizar la aplicación de forma sencilla y visualizar la información encontrada de la imagen reconocida.

Para reducir la cantidad de procesamiento, los datos necesarios para clasificar de las imágenes de la base de datos pueden ser extraídos con otro programa por separado, e incluir estos datos ya calculados en el paquete descargable. Para reducir el espacio en memoria que ocuparía una aplicación con una base de datos grande, la base de datos con los datos extraídos únicamente contiene un identificador de la imagen en cuestión, su descriptor según el método utilizado (basado en BOW o basado en la comparación de niveles de gris) y sus

puntos característicos y descriptores para poder hacer la homografía al final; otros datos como información concreta de la imagen o la imagen en sí pueden ser descargados de un servidor o visualizados en un navegador de Internet.

3.3.1. FUNCIONAMIENTO

Esta sección muestra el funcionamiento de la aplicación independientemente de cuál sea el método utilizado para el reconocimiento de imágenes. En la FIGURA 13 se muestran los pasos dados por la aplicación en un dispositivo móvil.

FIGURA 13. DIAGRAMA DE PASOS DEL FUNCIONAMIENTO DE LA APLICACIÓN.

1. Se comprueba si la librería de *OpenCV* está instalada en el dispositivo. Esta librería es una aplicación desarrollada por *OpenCV* que permite a los usuarios ejecutar aplicaciones que utilicen sus funciones. Si la aplicación no se encuentra instalada, se da la opción de ir a la *Google Play Store* para su descarga.
2. Se comprueba si la base de datos del dispositivo está en la misma versión del fichero con el que se distribuye. Si la base de datos no está creada, se genera basándose en este fichero. Si no está actualizada, se añaden o cambian los campos necesarios.

3. La propia *API* de Android, mediante el uso de *Intents*, permite a los desarrolladores abrir la cámara del dispositivo para hacer una foto o traerla de la galería.
4. Se corrige la perspectiva de la foto capturada y se calcula su descriptor basado en BOW o en la comparación de niveles de gris, sus puntos característicos y su descriptor usando ORB.
5. Se calculan las distancias del descriptor extraído con los de la base de datos y obtienen los diez más próximos.
6. A las diez imágenes más próximas, se les aplica la homografía y se da como imagen reconocida aquella con mayor porcentaje de *inliers*.
7. Con el identificador de la imagen reconocida se muestra la información relativa a ella. Por ejemplo, en el caso concreto del reconocimiento de cartas *"Magic"*, la información se descarga de un servidor de Internet (*DeckBrew.com*). Estos datos se reciben en formato JSON, que son interpretados en la aplicación y se muestran por pantalla.

4. PRUEBAS Y RESULTADOS

Para constatar el correcto funcionamiento de los métodos explicados, se construyen dos programas de prueba para medir el porcentaje de aciertos que devuelven y la eficiencia de estos métodos medida en tiempo de ejecución. Estos programas se ejecutan en un ordenador de sobremesa con estas características:

Sistema Operativo	Ubuntu 12.04 LTS
Procesador	*Intel Core i3* (3,07 GHz)
Memoria RAM	8 GB

TABLA 3. CARACTERÍSTICAS DEL ORDENADOR DONDE SE REALIZARON LAS PRUEBAS.

Tras las pruebas se decide qué método será el desarrollado para su uso en dispositivos móviles Android. Al final de este capítulo se encuentran pruebas de eficiencia y de usabilidad ya en un entorno móvil. Para estas pruebas se utiliza un dispositivo con las siguientes características:

Modelo	*Samsung Galaxy S 5* (G900F)
Procesador	*Snapdragon Qualcomm* (2,5 GHz)
Memoria RAM	2 GB
Memoria Interna	16 GB

TABLA 4. CARACTERÍSTICAS DISPOSITIVO MÓVIL UTILIZADO PARA LAS PRUEBAS.

Además, se incluyen varias capturas de pantalla de la aplicación en uso y cómo se notifican errores en el proceso al usuario.

4.1. PRUEBAS DE ACIERTO

4.1.1. BANCO DE IMÁGENES DE PRUEBA

Para la realización de las pruebas se ha utilizado la colección de cartas de "*Magic: The Gathering*", edición 2014. Se ha elegido esta colección para disponer de un número lo bastante grande de imágenes y tener todas las mismas dimensiones y formato.

Colección	*"Magic: The Gathering"*, edición M14
Dimensiones	480 x 680
Cantidad de imágenes	262
Formato	JPG

TABLA 5. CARACTERÍSTICAS IMÁGENES DE PRUEBA.

Estas imágenes han sido extraídas de la web *MTGimage.com*, que dispone de una base de datos con muchas de las imágenes de este juego de cartas en buena resolución.

De estas cartas se han impreso cien para después hacerles una fotografía con el móvil. Serán estas fotografías las que sirvan de entrada a los programas de prueba. Para simplificar el procedimiento, estas imágenes se han tomado todas con la misma resolución y dimensiones (1152 x 2048), sobre un fondo blanco y sin más objetos en la imagen para que no interfieran a la hora de la clasificación (ver FIGURA 14).

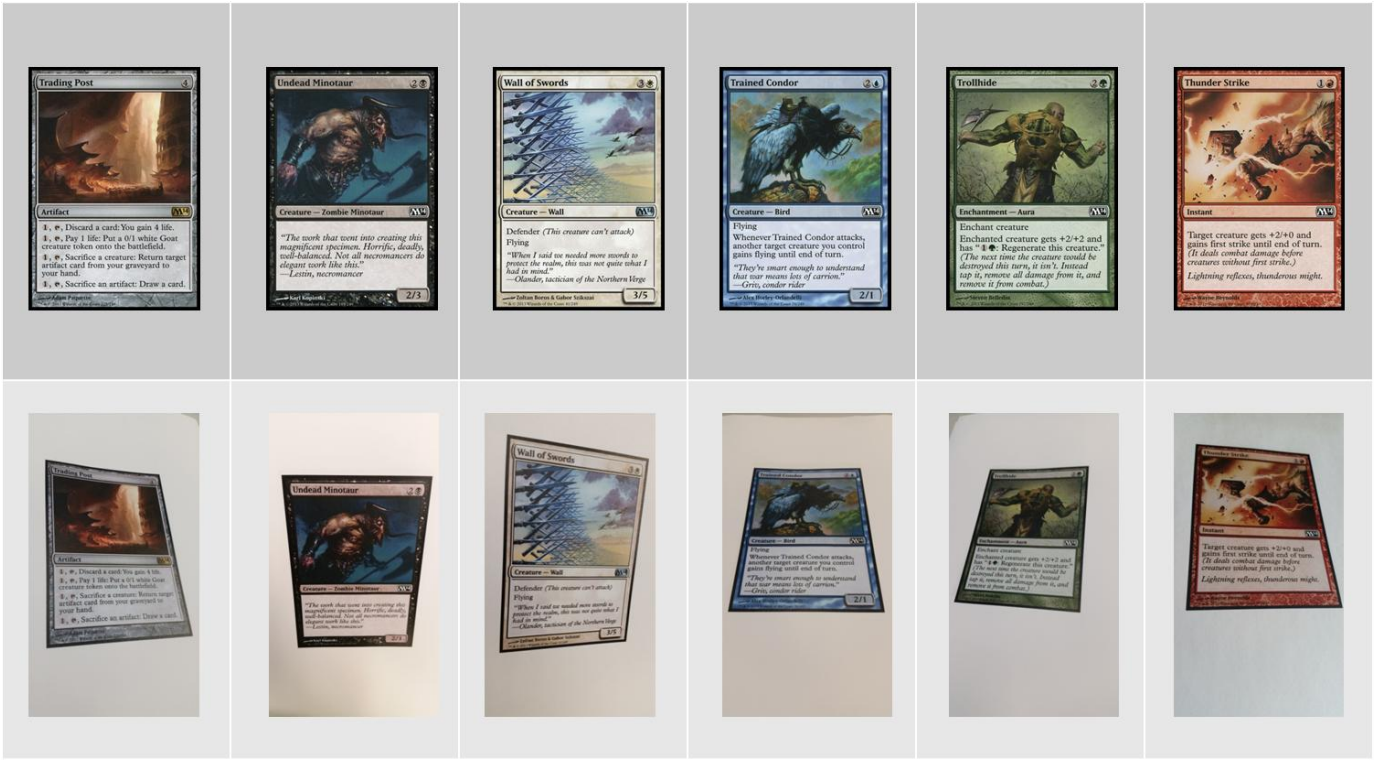


FIGURA 14. ARRIBA: IMÁGENES DE LA BASE DE DATOS. ABAJO: FOTOGRAFÍAS TOMADAS CON EL MÓVIL DE LAS MISMAS CARTAS.

4.1.2. PROCEDIMIENTO Y RESULTADOS

4.1.2.1. Bolsa de palabras (BOW)

4.1.2.1.1. Procedimiento

Como se describía en el apartado de diseño y desarrollo, primero se entrenan las imágenes de la base de datos para crear un diccionario que almacene las palabras extraídas de estas y su identificador.

Por cada imagen de entrada, se extrae su descriptor y se usa la función de vecinos más próximos para seleccionar las treinta y dos imágenes de la base de datos más cercanas según este método.

Se compara el identificador de la carta de entrada con los identificadores de las cartas seleccionadas, para saber si alguna de ellas es la correcta. En caso afirmativo, se observa su posición en el *ranking*. La posición de esta imagen en el *ranking* es una medida de la calidad del método.

Las pruebas se basan en cambiar el número de palabras del diccionario que usa BOW (aumentando de cincuenta en cincuenta, desde cincuenta hasta novecientos cincuenta), para tener un diferente número de palabras en el vocabulario utilizado. Se realizarán pruebas sobre cartas donde no se ha aplicado la corrección de perspectiva, con una extracción manual de la región de interés y con la corrección de perspectiva automática (ver FIGURA 15), para comprobar la importancia de estos hechos en el reconocimiento.

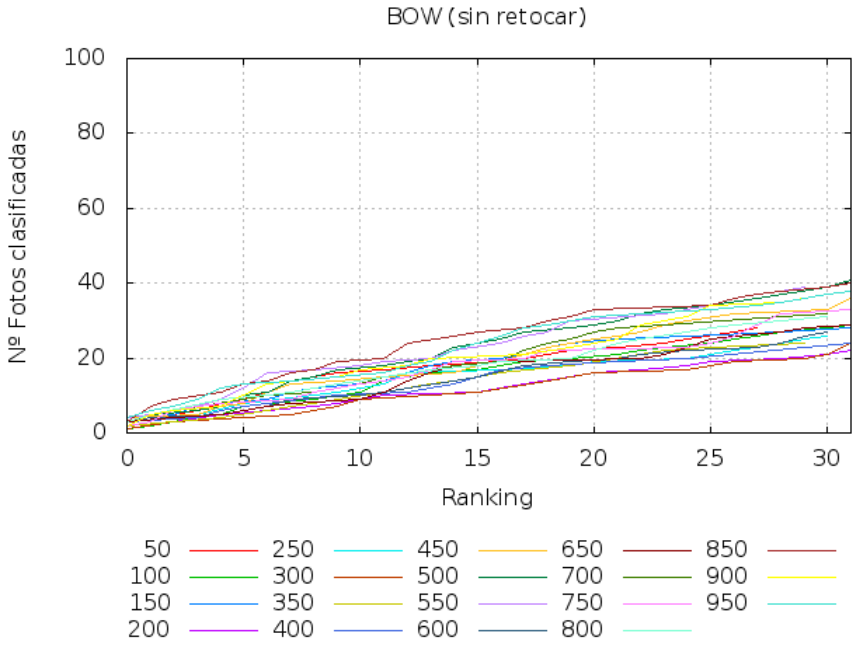
FIGURA 15. IZQUIERDA: IMAGEN TOMADA SIN RETOCAR. CENTRO: IMAGEN EN LA QUE SE HA APLICADO EL RECORTE MANUAL. IZQUIERDA: IMAGEN EN LA QUE SE HA APLICADO LA CORRECCIÓN AUTOMÁTICA DE PERSPECTIVA.

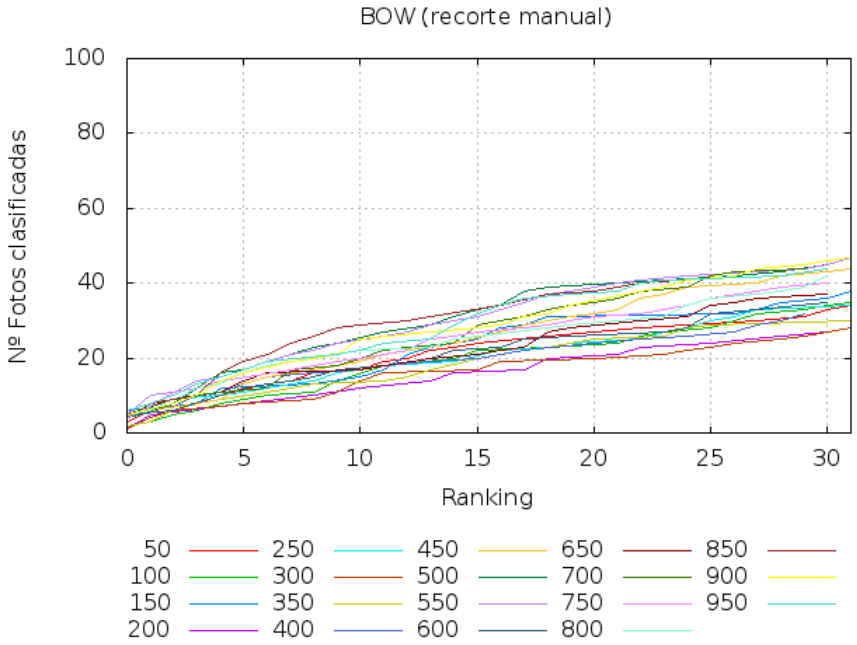
4.1.2.1.2. Resultados

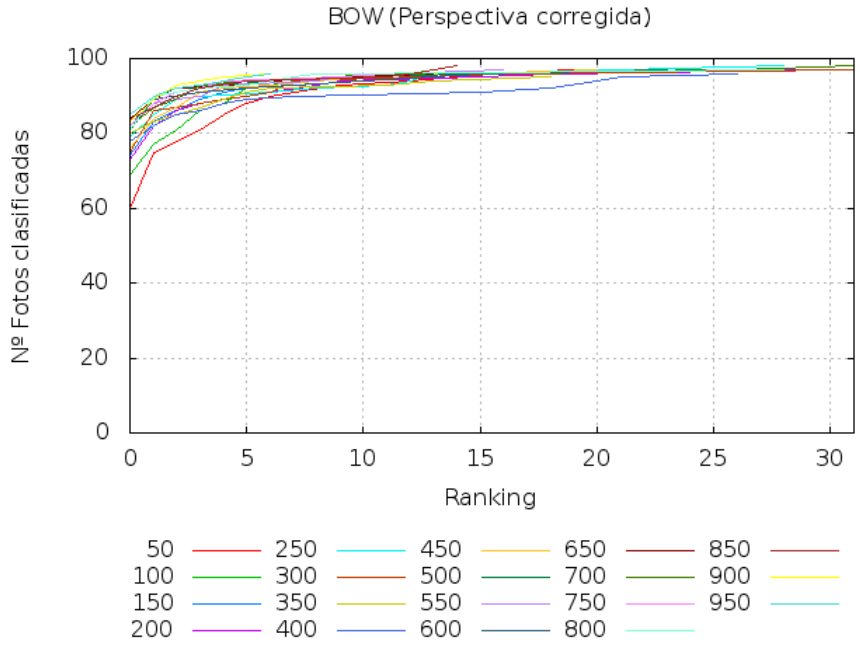
En el conjunto de GRÁFICA 1, se puede observar cómo las cien imágenes de entrada son clasificadas en un *ranking*. En este caso, se representa qué cantidad de fotografías han sido encontradas (eje Y) entre qué posiciones del ranking (eje X). Cada línea representa un tamaño de diccionario diferente.

Como se ve en las pruebas, el algoritmo basado en BOW mejora algo al hacerse un recorte manual del fondo y mucho al extraer la región de interés automáticamente. Esto a priori no se esperaba, pues el algoritmo de BOW se basa en puntos característicos y estos son invariables frente a rotaciones o cambios de tamaño. La mejora en los resultados se debe a que el algoritmo de extracción de puntos característicos, detecta puntos en los bordes de la imagen de entrada o en el fondo de la misma, obteniendo de ahí palabras que se incluyen en el vocabulario que no son compartidas por las imágenes que componen la base de datos.

En la gráfica con la corrección de perspectiva, se observa que el número de aciertos mejora mientras a medida que aumenta el taño del vocabulario. Se puede ver cómo el algoritmo clasifica sesenta de las imágenes en el *top 0* cuando se usan sólo cincuenta palabras en el diccionario, mientras que cuando se usan novecientas cincuenta palabras, el acierto en el *top 0* aumenta hasta ochenta y seis.


BOW (sin retocar)
Nº Fotos clasificadas
Ranking
0
20
40
60
80
100
0
5
10
15
20
25
30
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950


BOW (recorte manual)
Nº Fotos clasificadas
Ranking
0
20
40
60
80
100
0
5
10
15
20
25
30
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950



GRÁFICA 1. GRÁFICAS DE ACIERTO SEGÚN EL ESTADO DE LA IMAGEN DE ENTRADA. PRIMERA GRÁFICA: ENTRADA SIN RECORTAR. SEGUNDA GRÁFICA: RECORTE DE LA ZONA DE INTERÉS HECHO A MANO. TERCERA GRÁFICA: USANDO EL ALGORITMO DE CORRECCIÓN DE PERSPECTIVA.

4.1.2.2. Método basado en la comparación de niveles de gris en diferentes secciones de la imagen

4.1.2.2.1. Procedimiento

De la misma manera que la prueba anterior, usando este método, se extraen los descriptores de todas las imágenes de la base de datos, y se almacenan con su identificador.

Se extrae el descriptor de cada imagen de entrada y se compara con todos los de la base de datos usando la distancia de Hamming. Se ordena la base de datos en función a la distancia, de más baja a más alta, y se consigue la lista ordenada de imágenes similares. Con el objetivo de poder comparar las dos pruebas, sólo se tiene en cuenta el *top 32* de esta lista.

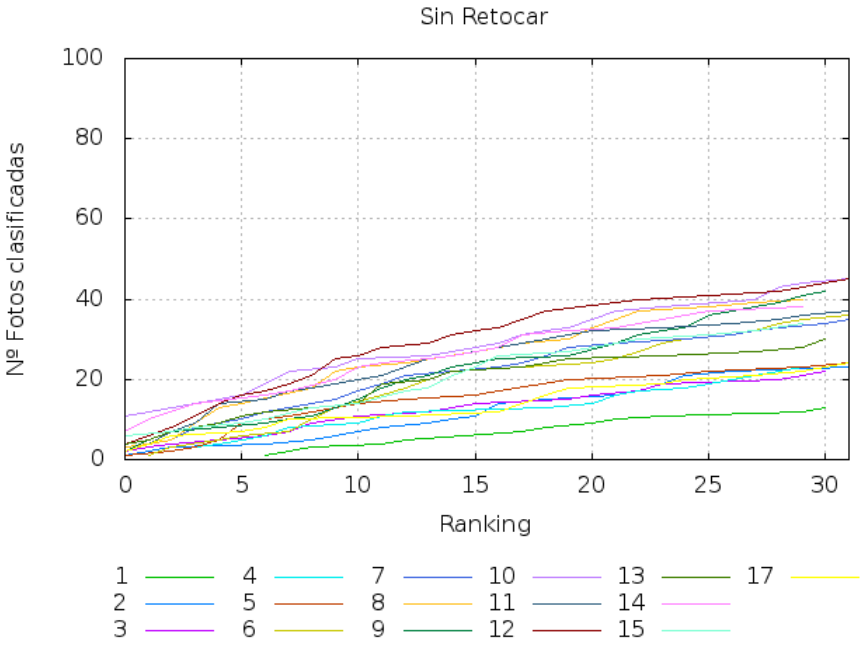
De nuevo, se repite la prueba para las tres colecciones de fotos ya mencionadas. En esta ocasión, lo que varía es la profundidad de recursión a la que se llega, subiendo el nivel de uno en uno hasta diecisiete.

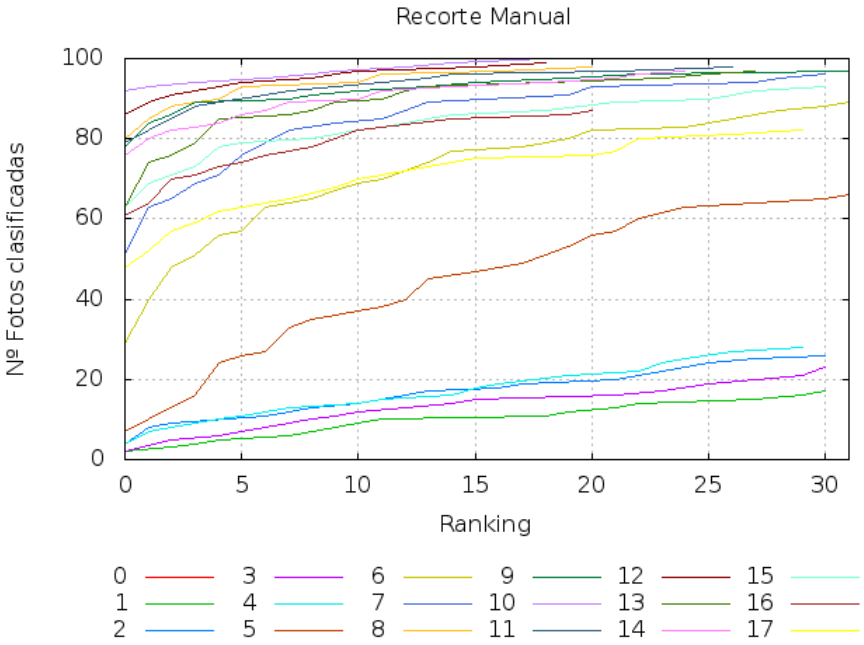
4.1.2.2.2. Resultados

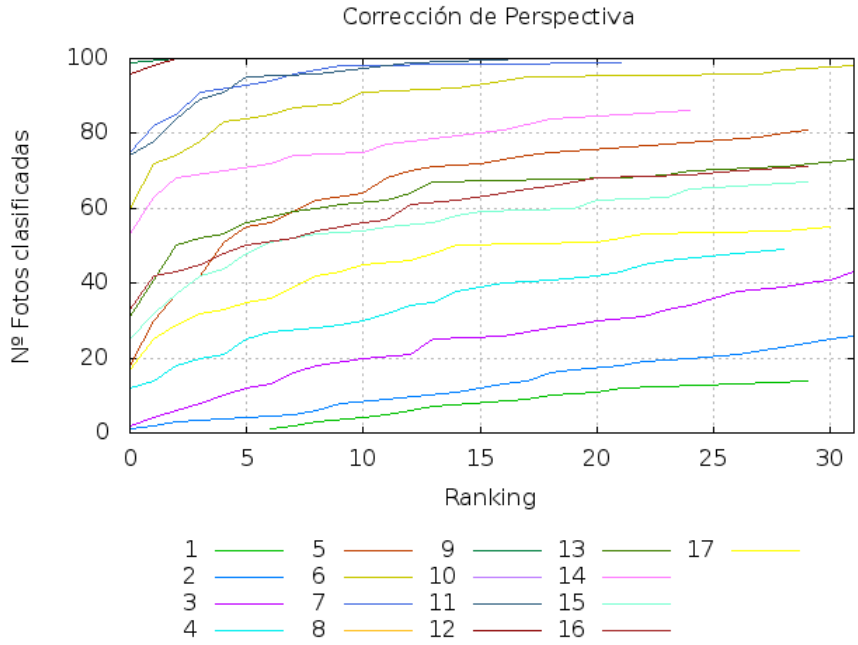
Esta vez, podemos observar en el conjunto de GRÁFICA 2 más variabilidad en función de a qué nivel de recursión se llegue con este método. Con la corrección automática de perspectiva, si se llega a un nivel entre el 7 y el 11, se puede clasificar una imagen en el *top 5* del *ranking* alrededor del 90% de las veces. Si el nivel de recursión al que se llega es mayor, este porcentaje decae, ya que el posible ruido que contenga la fotografía adquiere demasiada importancia.

También cabe recalcar que se observa mejora con el recorte manual de la región de interés, mejorando así a los resultados obtenidos con BOW en este mismo caso. La técnica aplicada en esta vez no es tan sensible a la aparición de elementos como el fondo o los bordes en la imagen, pues se compensa gracias a la estructura del descriptor.

Durante la ejecución de estas pruebas se ha visto una colocación en el *top 0* de noventa y nueve o cien imágenes con niveles de recursión 8, 9 y 10 usando corrección de perspectiva. Este hecho no puede darse en la práctica, pues puede haber errores en la fotografía que hagan que la imagen no sea bien reconocida.


Sin Retocar
Nº Fotos clasificadas
Ranking
0
20
40
60
80
100
0
5
10
15
20
25
30
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
17


Recorte Manual
Nº Fotos clasificadas
Ranking
0
20
40
60
80
100
0
5
10
15
20
25
30
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17



GRÁFICA 2. GRÁFICAS DE ACIERTO SEGÚN EL ESTADO DE LA IMAGEN DE ENTRADA. PRIMERA IMAGEN: ENTRADA SIN RECORTAR. SEGUNDA IMAGEN: RECORTE DE LA ZONA DE INTERÉS HECHO A MANO. TERCERA IMAGEN: USANDO EL ALGORITMO DE CORRECCIÓN DE PERSPECTIVA.

4.2. PRUEBAS DE EFICIENCIA

4.2.1. BANCO DE IMÁGENES DE PRUEBA

Al igual que en el apartado anterior, se usan las imágenes de la colección de "*Magic: The Gathering*" (edición 2014) para la creación de la base de datos. Las imágenes de entrada son las tomadas por el móvil y se le aplicará el algoritmo de corrección de perspectiva.

4.2.2. PROCEDIMIENTO Y RESULTADOS

4.2.2.1. Bolsa de palabras (BOW)

4.2.2.1.1. Procedimiento

Para esta prueba se ha usado el mismo programa que la prueba anterior, haciendo pasar todas las fotografías de entrada por el algoritmo de corrección de perspectiva. Se ha medido en milisegundos lo que se tarda en detectar sus puntos característicos, en extraer su descriptor según el diccionario creado y en calcular sus vecinos más próximos.

4.2.2.1.2. Resultados

En esta prueba se puede ver como el tiempo total es bastante constante independientemente del tamaño del diccionario (unos 40 segundos en clasificar 100 fotografías). Si analizamos los datos más profundamente, se puede ver que tanto la fase de extraer el descriptor, como la búsqueda de los vecinos más próximos aumentan levemente de forma lineal según aumenta el tamaño del diccionario. En la TABLA 6 se muestran los tiempos en el caso que consigue más clasificaciones en el primer lugar del *ranking* (con 950 palabras en el diccionario).

Tiempo en detectar los puntos característicos	11,47 s
Tiempo en extraer los descriptores BOW	28,07 s
Tiempo en calcular los vecinos más próximos	0,02 s

TABLA 6. TIEMPOS DE EJECUCIÓN DEL MÉTODO BASADO EN BOW CON 950 PALABRAS.

4.2.2.2. Método basado en la comparación de niveles de gris en diferentes secciones de la imagen

4.2.2.2.1. Procedimiento

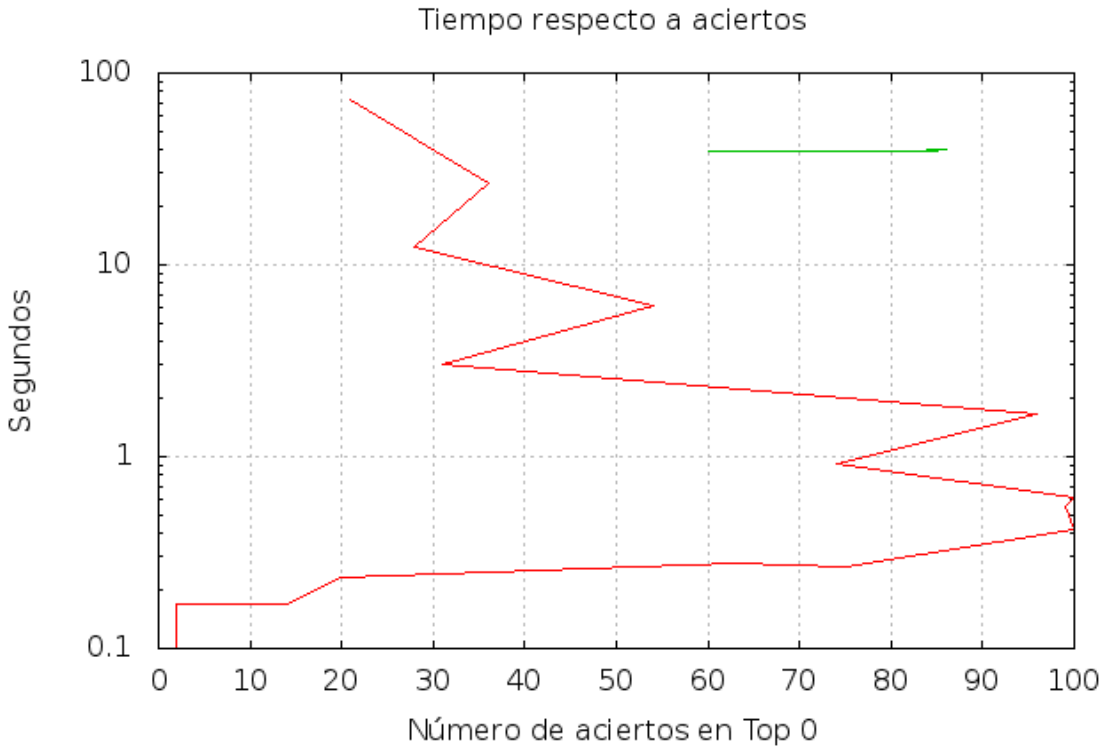
Se utiliza el mismo programa de prueba que en el apartado anterior, haciendo corregir la perspectiva de las fotos de entrada en todos los casos. Se mide en milisegundos el tiempo que se tarda en corregir la perspectiva de la imagen de entrada, en extraer su descriptor y en clasificar la imagen.

4.2.2.2.2. Resultados

Se ha comprobado el tiempo de ejecución del algoritmo basado en la comparación de niveles de gris para el nivel 10 de recursión, que es el que más clasificaciones consigue en el *top 0* del *ranking*: se tarda unos 0,6 segundos en clasificar las cien imágenes de entrada. Este dato aumenta de forma exponencial según aumenta el nivel de recursión.

4.3. CONCLUSIONES SOBRE LAS PRUEBAS

A pesar de que la clasificación usando BOW tenga un tiempo constante de ejecución, el uso de la técnica basada en la comparación de niveles de gris tiene un tiempo de ejecución mucho menor cuando su porcentaje de aciertos es mayor. En la GRÁFICA 3 se exponen los tiempos para ambos métodos en función del número de aciertos en el *top 0*. En ella se observa como el tiempo de ejecución para BOW se mantiene constante mientras que el tiempo del otro método depende del nivel de recursión. También se observa como la técnica basada en comparación de niveles de gris consigue un acierto mayor y en menor tiempo para algunos niveles de recursión.



GRÁFICA 3. GRÁFICA DONDE SE MUESTRAN LOS ACIERTOS EN *TOP 0* SEGÚN EL TIEMPO DE EJECUCIÓN. EN ROJO: ALGORITMO BASADO EN LA COMPARACIÓN DE NIVELES DE GRIS. EN VERDE: ALGORITMO BASADO EN BOW.

Este hecho hace del método basado en la comparación de niveles de gris en diferentes secciones de la imagen el mejor de estos dos para implementar en un dispositivo móvil, ya que aparte de ser más rápido, es más certero en sus clasificaciones.

Como se mencionó al principio del documento, también se han hecho pruebas con una colección de portadas de libros. Los resultados obtenidos son parecidos, confirmando que la aplicación desarrollada funciona con varias bibliotecas de imágenes.

4.4. RESULTADO EN DISPOSITIVO MÓVIL

4.4.1. IMPLEMENTACIÓN EN EL MÓVIL

Como ya se comentó en la sección 3, junto a la aplicación se distribuye un fichero que contiene los descriptores basados en la comparación de niveles de gris de las imágenes de la base de datos, así como sus puntos característicos y descriptores ORB y un identificador.

4.4.2. PRUEBAS DE EFICIENCIA

Para comprobar la eficiencia de la aplicación en un entorno móvil, se ha probado a clasificar cincuenta imágenes y a sacar una media de tiempos de ejecución.

Tiempo Total	4,43 s
Tiempo en obtener el descriptor basado en la comparación de niveles de gris	0,1 s
Tiempo en extraer los puntos característicos	0,04 s
Tiempo en calcular los descriptores	0,04 s
Tiempo en comparar las distancias	2,47 s
Tiempo en hacer la homografía y clasificar la imagen	1,77 s

TABLA 7. TIEMPO MEDIO DE EJECUCIÓN DE LA APLICACIÓN EN UN DISPOSITIVO MÓVIL

4.4.3. INTERFAZ Y EJEMPLOS

En la FIGURA 16 se muestra el proceso de la aplicación cuando una imagen ha sido reconocida.

FIGURA 16. INTERFAZ GRÁFICA DE LA APLICACIÓN. PRIMERA CAPTURA: SELECCIÓN SOBRE LA FUENTE DE LA IMAGEN (CON LA CÁMARA O DESDE LA GALERÍA). SEGUNDA CAPTURA: LA FOTO HA SIDO SELECCIONADA Y SE ESTÁ RECONOCIENDO. TERCERA CAPTURA: LA IMAGEN HA SIDO RECONOCIDA Y SE MUESTRAN SUS DATOS.

En la FIGURA 17 se muestra la pantalla cuando ha ocurrido un error, ya sea al tomar la foto (no se ha podido corregir la perspectiva correctamente) o cuando la imagen no se encuentra en la base de datos.

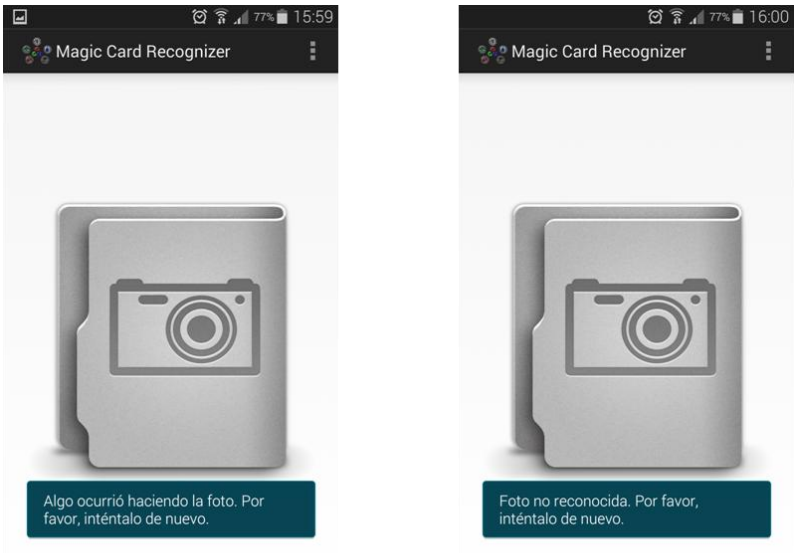


FIGURA 17. IZQUIERDA: ERROR AL HACER LA FOTOGRAFÍA O AL CORREGIR LA PERSPECTIVA. DERECHA: IMAGEN DE ENTRADA NO RECONOCIDA.

5. CONCLUSIONES

Desde la aparición de los teléfonos inteligentes y tabletas, ha cambiado la forma en la que interactuamos con el mundo. Proyectos abiertos como Android y *OpenCV* hacen que todos podamos contribuir al desarrollo de nuevas aplicaciones y paradigmas. El reconocimiento de imágenes y su distribución en dispositivos móviles ofrece un gran abanico de posibilidades que aún está por explotar, ya que todavía hay que mejorar los algoritmos y adaptarlos a nuevas necesidades y situaciones a las que nos enfrentamos.

5.1. TRABAJO DESARROLLADO Y FUTURO

Este trabajo de final de grado tenía como objetivo el desarrollo de una aplicación para dispositivos Android que fuese capaz de mostrar información sobre un objeto bidimensional con sólo hacerle una fotografía. Para ello se han estudiado diferentes casos ya desarrollados tanto en el mundo comercial (p. ej. *Google Goggles* (1)) como en el académico (p. ej. el uso de puntos característicos (4) o BOW (2)).

Se han implementado dos alternativas diferentes: una basada en BOW y otra en descriptores globales que usaban la comparación de niveles de gris entre diferentes secciones de la imagen. Tras las pruebas, se ha constatado que la

alternativa que usa descriptores globales es más efectiva y rápida, por lo que es esta la que se ha llevado a la aplicación final.

Esta aplicación cumple con los objetivos planteados en la primera sección del documento (ver TABLA 1): el usuario puede tomar una fotografía con su dispositivo, y es dentro de éste mismo donde la imagen es reconocida. Para el ejemplo concreto que se ha implementado, la biblioteca de imágenes ha sido de cartas del juego *"Magic: The Gathering"* (Edición 2014) y los datos mostrados son descargados de un servidor externo. Simplemente cambiando la base de datos por otra cuyas imágenes sean rectangulares, la aplicación es capaz de reconocer otro tipo de objetos, como pueden ser portadas de libros o discos.

En un futuro, queda abierta la posibilidad de implementar mejoras en la aplicación que mejoren su rendimiento y usabilidad:

- Se podría mejorar el algoritmo de reconocimiento de la región de interés, para que pueda haber varios objetos en la imagen tomada o reconocer objetos con formas diferentes.
- Aumentar el tamaño de la base de datos de la aplicación para poder manejar colecciones más grandes, cuidando que el tiempo de ejecución siga siendo aceptable y el porcentaje de aciertos alto.
- Desarrollar nuevas aplicaciones aplicando los métodos estudiados y cambiando la interfaz con el usuario. Por ejemplo, se podría extraer la imagen a reconocer usando la cámara en modo vídeo y mostrando

la información superpuesta en la imagen, en vez de tener que tomar una fotografía y cambiar de vista.

5.2. CONCLUSIONES PERSONALES

Este proyecto ha abarcado un gran número de tecnologías; a parte del desarrollo de código en *C/C++* o *Java*, se ha tenido que consultar multitud de referencias, aprender *OpenCV* y su implementación para Android o descargar información de servidores externos utilizando *JSON*.

En mi opinión, ha resultado un proyecto muy interesante que contiene un alto potencial. Trabajando en él, he podido aprender muchos nuevos conceptos y aplicarlos a un campo que ya conocía (desarrollo de aplicaciones Android), que probablemente de otra manera no me habría planteado.

6. REFERENCIAS

1. **Google Inc.** Google Goggles. [En línea] https://play.google.com/store/apps/details?id=com.google.android.apps.unveil.

2. *Scalable Recognition with a Vocabulary Tree.* **Nistér, David y Stewénius, Henrik.** University of Kentucky, EEUU : s.n., 2006.

3. *Small Codes and Large Image Databases for Recognition.* **Torralba, Antonio, Fergus, Rob y Weiss, Yair.** MIT, EEUU; NYU, EEUU; Hebrew University, Israel : s.n., 2008.

4. *Planar Object Detection using Local Feature Descriptors.* **Kottman, Michal.** Slovak University of Technology, Bratislava, Slovakia : s.n., 2011.

5. *ORB: an efficient alternative to SIFT or SURF.* **Rublee, Ethan, y otros.** Willow Garage, Menlo Park, California : s.n., 2011.

6. **Wizards Of The Coast LLC.** "Magic: The Gathering", edición 2014 (M14). [En línea] http://gatherer.wizards.com/Pages/Search/Default.aspx?output=spoiler&method=visual&set=%5b%22Magic+2014+Core+Set%22%5d.

7. **Selmeier, Bill.** *Spreading the Barcode.* 2008.

8. **Furht, Borko.** *Handbook of Augmented Reality.* 2011.

9. **DroidLa.** QR Droid. *Google Play Store.* [En línea] https://play.google.com/store/apps/details?id=la.droid.qr&hl=es.

10. **Scanbuy Inc.** BIDI: lector QR y de barras. *Google Play Store.* [En línea] https://play.google.com/store/apps/details?id=com.bidi&hl=es.

11. **Google Inc.** Goggles overview and requirements. *Google Support.* [En línea] https://support.google.com/websearch/answer/166331.

12. **Adee, Sally.** Google: "Goggles Does NOT Do Face Recognition". *IEEE Spectrum.* [En línea] 2010. http://spectrum.ieee.org/tech-talk/consumer-electronics/portable-devices/google-goggles-does-not-do-face-recognition.

13. **Image Searcher Inc.** CamFind - Visual Search Engine. *Google Play Store.* [En línea]
https://play.google.com/store/apps/details?id=com.msearcher.camfind&hl=es.

14. **Bandai America.** NARUTO CARD SCANNER. *Google Play Store.* [En línea]
https://play.google.com/store/apps/details?id=com.bandai.naruto.cardscanner.

15. —. Scanner cartas Power Rangers. *Google Play Store.* [En línea]
https://play.google.com/store/apps/details?id=com.bandai.powerrangers.cardscanner.eurasia.

16. **Moodstocks.** Moodstocks. [En línea] https://moodstocks.com/.

17. **TinEye.** TinEye. [En línea] http://tineye.com/.

18. *Visual Categorization with Bags of Keypoints.* **Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, Cédric Bray.** Xerox Research Centre Europe : s.n., 2004.

19. **OpenCV.** Features2D + Homography to find a known object (Tutorial). *OpenCV Documentation.* [En línea]
http://docs.opencv.org/doc/tutorials/features2d/feature_homography/feature_homography.html.

20. *Distinctive Image Features from Scale-Invariant Keypoints.* **Lowe, David G.** University of British Columbia, Vancouver, B.C., Canada : s.n., 2004.

21. *SURF: Speeded Up Robust Features.* **Bay, Herbert, Tuytelaars, Tinne y Van Gool, Luc.** ETH Zurich; Katholieke Universiteit Leuven : s.n., 2006.

22. *BRIEF: Binary Robust Independent Elementary Features.* **Calonder, Michael, y otros.** Lausanne, Switzerland : s.n., 2010.

23. *KAZE Features.* **Alcantarilla, Pablo Fernández, Bartoli, Adrien y Davison, Andrew J.** Universit´e d'Auvergne, Clermont Ferrand, France; Imperial College London, UK : s.n., 2012.

24. *Mobile Product Recognition.* **Tsai, Sam S., y otros.** Stanford University, EEUU : s.n., 2010.

25. *Universal and Adapted Vocabularies for Generic Visual Categorization.* **Perronnin, Florent.** Xerox Research Centre Europe : s.n., 2007.

26. **Bradski, Gary y Kaehler, Adrian.** *Learning OpenCV.* s.l. : O'Reilly Media, 2008.

27. **OpenCV.** OpenCV. [En línea] http://opencv.org/.

28. **Elgin, Ben.** Google Buys Android for Its Mobile Arsenal. *Bloomberg Business Week.* [En línea] http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal.

29. **Google Inc.** Android. [En línea] http://www.android.com/.

30. **Open Handset Alliance.** Industry Leaders Announce Open Platform for Mobile Devices. *Open Handset Alliance.* [En línea] http://www.openhandsetalliance.com/press_110507.html.

31. **Google Inc.** Android Developer. [En línea] http://developer.android.com/.

32. *A Computational Approach To Edge Detection.* **Canny, J.** s.l. : IEEE Trans. Pattern Analysis and Machine Intelligence, 1986.

33. *A combined corner and edge detector.* **Harris, C. y Stephens, M.** s.l. : Proceedings of the 4th Alvey Vision Conference, 1988.

34. *Overview of the RANSAC Algorithm.* **Derpanis, Konstantinos G.** 2010.

35. **OpenCV.** Función CvtColor. *Documentación OpenCV.* [En línea] http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html#cvtcolor.

36. —. Understanding k-Nearest Neighbour. *Documentación OpenCV.* [En línea] http://docs.opencv.org/trunk/doc/py_tutorials/py_ml/py_knn/py_knn_understanding/py_knn_understanding.html.

37. **Schantz, Herbert F.** *The history of OCR, optical character recognition.* 1982.

38. *Machine learning for high-speed corner detection.* **Rosten, Edward y Drummond, Tom.** Cambridge University, UK : s.n., 2006.

39. *Tagging Products using Image Classification.* **Tomasik, Brian, Thiha, Phyo y Turnbull, Douglas.** Swarthmore College, EEUU : s.n., 2009.

GLOSARIO

- **BOW** (2): *Bag Of Words*, bolsa de palabras. Técnica usada en minería de datos e inteligencia artificial.
- **SIFT** (6): *Scale-Invariant Feature Transform*, transformación de puntos característicos invariantes a la escala. Tipo de punto característico.
- **SURF** (7): *Speeded Up Robust Features*, puntos característicos robustos acelerados. Tipo de punto característico.
- **ORB** (5): *Oriented FAST, Rotated BRIEF*, FAST orientado, BRIEF rotado. Tipo de punto característico.
- **FAST** (8): *Features from Accelerated Segment Test*, puntos característicos de segmento acelerado. Tipo de punto característico.
- **BRIEF** (9): *Binary Robust Independent Elementary Features*, puntos característicos elementales, independientes, binarios y robustos. Tipo de punto característico.

ANEXOS

ANEXO A. *MTG MAGIC CARD RECOGNIZER*

Escanee este BIDI con su dispositivo Android para poder descargar e instalar la aplicación de reconocimiento de cartas *"Magic: The Gathering"* (Edición 2014).

http://bit.ly/1mMYetz

Nota: Cuando esta aplicación es iniciada, comprueba que la librería de *OpenCV* está instalada en el dispositivo. Si no está instalada permite abrir la *Google Play Store* y descargar la aplicación necesaria.

ANEXO B. MÉTODOS UTILIZADOS

Conversión de una imagen a color a escala de grises

Las fotografías digitales tomadas a color están formadas de píxeles, que se componen de tres valores diferentes para el color rojo, verde y azul, que son los llamados canales R, G y B. En *OpenCV*, una fotografía se compone de una matriz tridimensional compuesta por estos tres canales combinados para poder ser mostrados.

El paso a escala de grises se hace aplicando la siguiente fórmula a los canales y devolviendo una matriz bidimensional, donde cada pixel es el resultado de su aplicación en los mismos pixeles de los tres canales (35):

$$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}144 \cdot B$$

Distancia de Hamming

La distancia de Hamming entre dos cadenas de la misma longitud se calcula comparando los elementos de cada cadena uno a uno. Si los elementos son diferentes, la distancia aumenta en uno.

Por ejemplo:

- La distancia entre “123456” y “023406” es 2
- La distancia entre “0110010” y “1111111” es 4

k-NN (K nearest neighbours, vecinos más próximos)

kNN (36) es uno de los algoritmos de clasificación más simples que existen. La idea se basa en considerar un espacio en el que se encuentran puntos de diferentes clases. Al añadir un nuevo punto al espacio, se seleccionan los *k* puntos más cercanos, y el nuevo punto es clasificado dentro de la clase a la que pertenezcan más de estos puntos seleccionados. En caso de empate, el nuevo punto sería clasificado con el más cercano. En el programa de prueba desarrollado con BOW, se usa este método para clasificar una imagen de entrada dentro de una clase que es una imagen de la base de datos. Se extrae una lista de los treinta y dos más cercanos porque sólo hay un punto por imagen de la base de datos y es interesante comprobar cuales son las imágenes más cercanas.

K-means

Es un método usado en minería de datos para agrupar varias observaciones (puntos) en *k* grupos en el que cada observación pertenece al grupo más cercano a la media. Primero, se incluyen aleatoriamente *k* puntos llamados centroides en el espacio de puntos que se va a agrupar. Se calcula qué centroide es el más cercano a cada punto y se mueve el centroide a la media de esta primera agrupación. Se vuelve a calcular cuales son los más cercanos y se vuelve a mover el centroide un número determinado de veces o hasta que los centroides no cambien (converjan) o se muevan menos que un umbral definido. Esta táctica se usa en este trabajo para agrupar los puntos característicos en grupos que serán las palabras del vocabulario de BOW.